



# Android应用开发

主讲教师 胡鑫

[hithuxin@hit.edu.cn](mailto:hithuxin@hit.edu.cn)

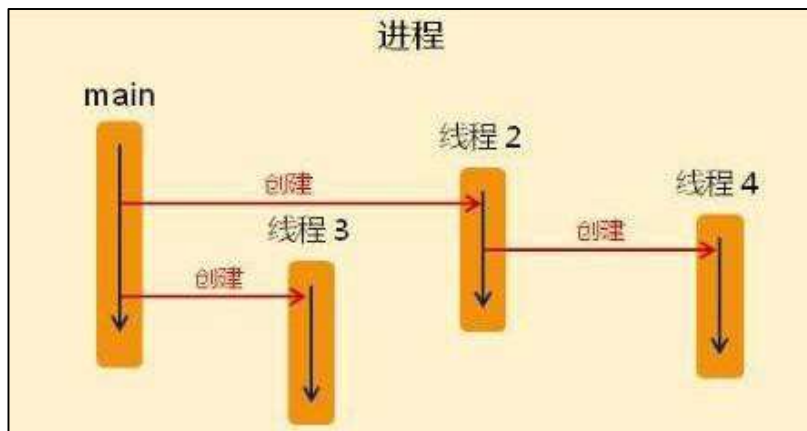
## 第11章 多线程和消息处理机制

1. [线程基本用法](#)
2. [Handler消息处理机制](#)
3. [ProgressBar](#)
4. [SeekBar](#)
5. [ListView+SeekBar音乐播放器](#)

## 11.1 线程基本用法

### ■ 基本概念：

- **进程**：一个正在系统上运行的程序。
- 一个进程包含一到多个**线程**。
- 多线程目的：实现多个线程**并发**执行，提高系统利用效率。



## ■ 线程定义方法1：继承Thread类

```
class MyThread extends Thread {  
    public void run() {  
        // 执行耗时操作等  
    }  
}
```

使用：

```
new MyThread().start();
```

## ■ 线程定义方法2：实现Runnable接口

```
class MyThread implements Runnable {  
    @Override  
    public void run() {  
        // 执行耗时操作等  
    }  
}
```

使用：

```
MyThread mythread=new MyThread();  
new Thread(mythread).start();
```

## ■ 常用简化形式：匿名类写法



```
new Thread( new Runnable() {  
    @Override public  
    void run() {  
        //执行耗时操作等  
    }  
}).start();
```

进一步简化

```
new Thread( ) {  
    @Override  
    public void run() {  
        //执行耗时操作等  
    }  
}.start();
```

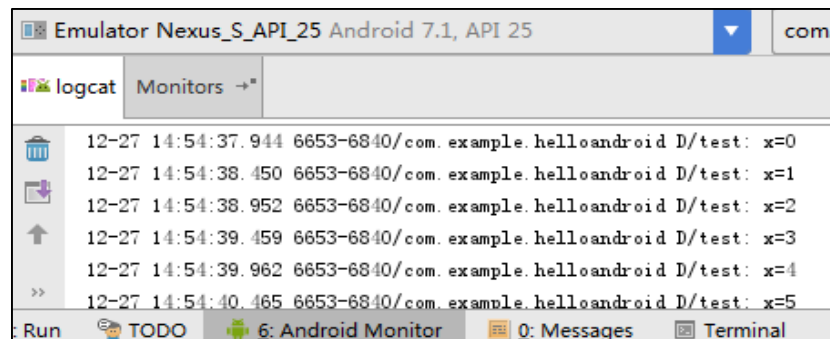
## 多线程示例

定义成员 `private int x=0;`



```
final TextView tv=(TextView)findViewById(R.id.msg);
Button bt2=(Button)findViewById(R.id.btn_test2);
bt2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                while (x<100)
                {
                    Log.d("test", "x=" + x++);
                    try {
                        Thread.sleep(500); //休息500ms继续
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }).start();
    }
});
```

```
Button bt1=(Button)findViewById(R.id.btn_test1);
bt1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        tv.setText("x="+x);
    }
});
```



## 修改前面例子：在子线程中修改UI（红色代码）

```
bt2.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        new Thread(new Runnable() {  
            @Override  
            public void run() {  
                while (x<100) {  
                    Log.d("test", "x=" + x++);  
                    tv.setText("x="+x);  
                    try {  
                        Thread.sleep(500);  
                    } catch (InterruptedException e) {  
                        e.printStackTrace();  
                    }  
                }  
            }  
        }).start();  
    }  
});
```

导致异常

试一下：通过  
Thread.currentThread().getName() 显  
示一下当前线程的名称



## 一个重要问题

- UI的更新**必须在主线程**（UI线程）中进行，如果在子线程中更新 UI 会导致异常。

```
----- beginning of crash
E/AndroidRuntime: FATAL EXCEPTION: Thread-4
    Process: com.example.helloandroid, PID: 13176
    android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.
        at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:6891)
        at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1048)
```

## 11.2 Handler消息处理机制

- 不直接在子线程中进行 UI 操作。
- 子线程通过 **Handler** 将**Message** 传送出去。
- **主线程中的Handler**接收这个Message，然后进行 UI 等相关操作。

子线程如何与主线程通信??

导入相关包：

```
import android.os.Handler;  
import android.os.Message;
```

注意：在Android Studio中，第一次创建Handler时默认导入的是 `import java.util.logging.Handler`

## 消息处理机制



主线程中创建Handler(作为Activity的一个成员变量)

```
private Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        if (msg.what == 某个消息标识) {
            // 处理消息
        }
    }
};
```

子线程中用Handler发送消息

```
new Thread( new Runnable() {
    @Override
    public void run() {    //实现run方法
        // 子线程完成某耗时操作
        // 发送消息
    } //end run
}).start();    //启动线程
```

推荐用: Message m = Message.obtain();

Message对象简单传值.arg1, .arg2, .obj等,  
也可以通过setData(bundle)传值

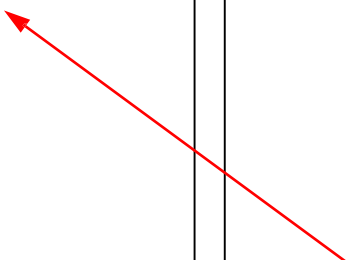
如果不需要传值, 简单使用  
**mHandler.sendMessage(消息标识即what值)**

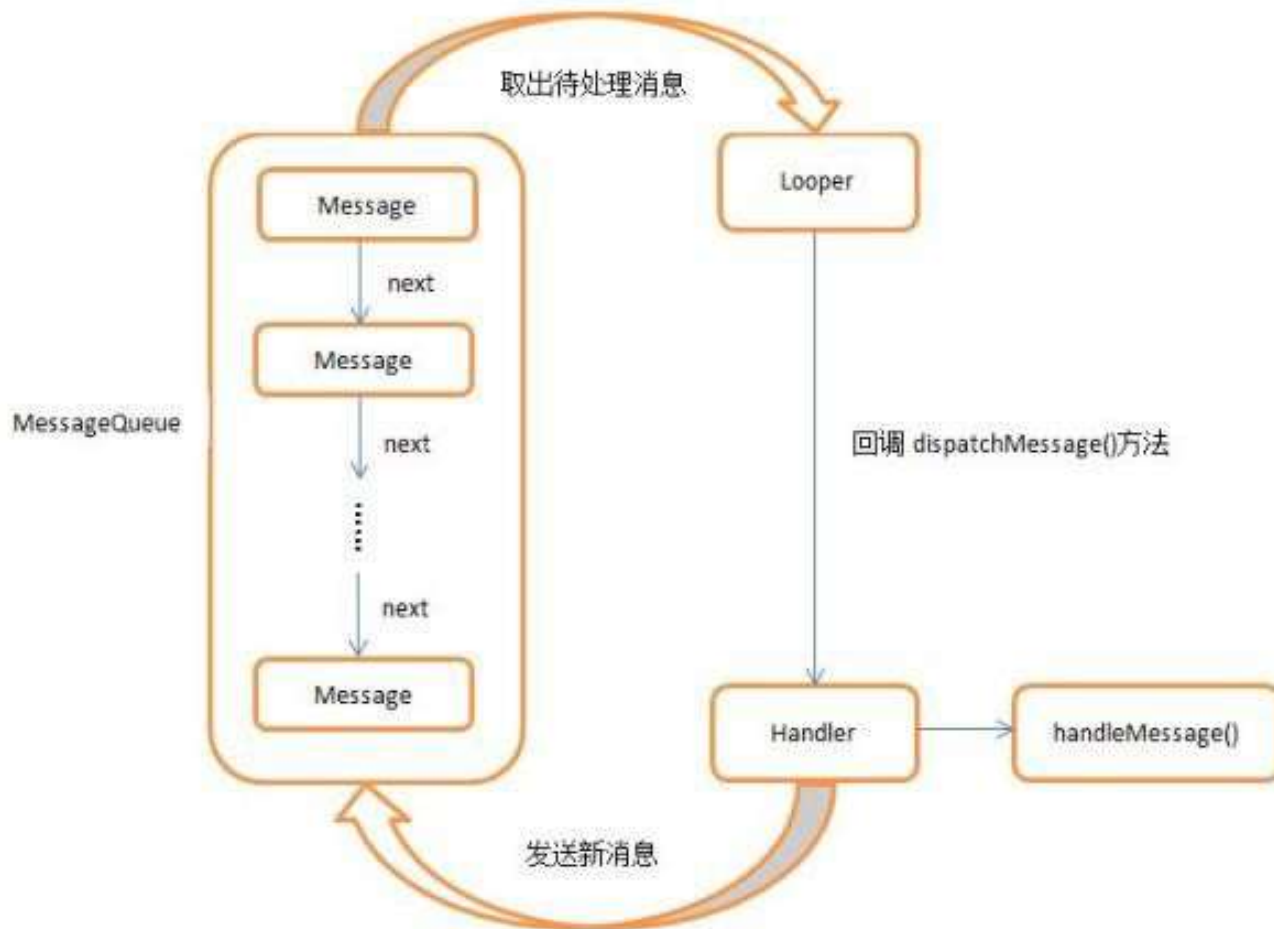
```
Message m = new Message();  
m.what = int值; //消息标识码  
m.arg1 = int值 //要传递的值如进度  
// 发送消息到指定Handler  
mHandler.sendMessage(m);
```

## 消息处理示例

```
private Handler mHandler=new Handler() {  
    @Override  
    public void handleMessage(Message msg) {  
        if( msg.what==123 ){  
            tv.setText("x="+ msg.arg1 );  
        }  
    }  
};
```

```
new Thread(new Runnable() {  
    @Override public  
    void run() {  
        while (x<100) {  
            Log.d("test", "x=" + (x++) );  
            try {  
                Thread.sleep(500);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
            Message message=new Message();  
            message.what=123;  
            message.arg1=x;  
            mHandler.sendMessage(message);  
        }  
    }  
}).start();
```





补充：

- 在子线程中，可以使用主线程创建的Handler的 `post(Runnable)` 方法，将Runnable钩到主线程中运行。

```
Button bt=(Button)findViewById(R.id.btn_test);  
bt.setOnClickListener(new View.OnClickListener() {
```

```
    @Override  
    public void onClick(View v) {  
        //线程代码  
    }
```

```
});
```

Handler.post(Runnable)可以把子线程中的UI操作送到主线程中完成

```
new Thread(new Runnable() {  
    @Override  
    public void run() {  
        while (x<100){  
            Log.d("test", "x=" + (x++) );  
            mHandler.post(new Runnable() {  
                @Override  
                public void run() {  
                    tv.setText("x="+x);  
                }  
            });  
            try {  
                Thread.sleep(500);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}).start();
```

【[返回](#)】

## 11.3 ProgressBar

### ■ ProgressBar基本属性：

```
<ProgressBar  
    android:id="@+id/progressBar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    style="@android:style/Widget.DeviceDefault.Light.ProgressBar" />
```

设置样式



## ProgressBar示例：显示与隐藏

```
Button btn1=(Button)findViewById(R.id.button);
btn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ProgressBar p = (ProgressBar) findViewById(R.id.progressBar);
        if (p.getVisibility() == View.GONE) {
            p.setVisibility(View.VISIBLE);
        }
        else {
            p.setVisibility(View.GONE);
        }
    }
});
```

## 水平ProgressBar

### ■ ProgressBar基本属性：

```
<ProgressBar
    android:id="@+id/progressBar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"

    android:progress="0" ← 设置进度值
    android:max="100" ← 设置进度条最大值
    style="@android:style/Widget.DeviceDefault.Light.ProgressBar.Horizontal"
/> ← 设置样式
```

## 水平ProgressBar示例1

```
Button btn1=(Button)findViewById(R.id.button);
btn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ProgressBar p = (ProgressBar) findViewById(R.id.progressBar);
        int progress = p.getProgress();
        progress += 10;
        p.setProgress(progress);
    }
});
```

## 练习：

- 每隔0.5秒让进度条进程+10，直到100结束。
- 思路：
  - 子线程：当 $\text{progress} < 100$ 时，  $\text{progress} + 10$  然后通过消息机制将 $\text{progress}$ 值传到主线程 子线程休息0.5秒继续。
  - 主线程：从消息中获取 $\text{progress}$ 值，更新进度条。

## 水平ProgressBar示例2

示例： 以填充长度为100的数组，模拟一个耗时操作进度

点击启动按钮之后，启动一个新的(子)线程来执行耗时任务，主线程通过Handler来接收子线程传回消息Message来更新滚动条。

```

public class MyActivity extends Activity {
    private int[] data = new int[100];    //填充长度为100的数组
    private int hasData = 0;    //数组已经填充的元素个数
    ProgressBar p;

    private Handler mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            if (msg.what == 0x111) {    //判断消息标识
                int progress=msg.arg1;    //取出子线程传来的值
                p.setProgress(progress);
                Log.d("test", "progress=" + progress);
                if (progress >= 100)
                    Toast.makeText(getApplicationContext(), "完成", Toast.LENGTH_SHORT).show();
            }
        }
    };
};
    
```

定义Handler  
来处理消息

//以填充数组来模拟一个耗时的操作

```
public int doWork(){
```

```
    data[hasData++] = (int)(Math.random() * 100); //填充一个数组元素
```

```
    try{
```

```
        Thread.sleep(100); //线程休息100毫秒，用于模拟其他耗时动作
```

```
    }
```

```
    catch (InterruptedException e){
```

```
        e.printStackTrace();
```

```
    }
```

```
    return hasData; //返回已填充的个数，亦即进度值
```

```
}
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
    p = (ProgressBar) findViewById(R.id.progressBar);
    Button btn1=(Button)findViewById(R.id.button);
    btn1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            hasData=0; //重新开始
            //启动线程，代码见后页
        } //end onClick
    }); // end setOnClickListener
} //end onCreate
} // end class
```



## 子线程

```
new Thread(new Runnable() { //新建一个(子)线程
    @Override
    public void run() { //实现 run() 方法，完成耗时操作
        while (hasData < 100){
            int progress = doWork(); //获取的是耗时操作的完成百分比
            Message m = new Message(); //创建消息
            m.what = 0x111; //设置消息标识
            m.arg1=progress; //简单传值
            mHandler.sendMessage(m); // 发送消息到Handler
        } //end while
    } //end run
}).start(); //启动线程
```

创建消息

## 11.4 SeekBar

### ■ SeekBar基本属性：

```
<SeekBar  
    android:id="@+id/seekBar"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:progress="0"/>
```

## SeekBar改变事件

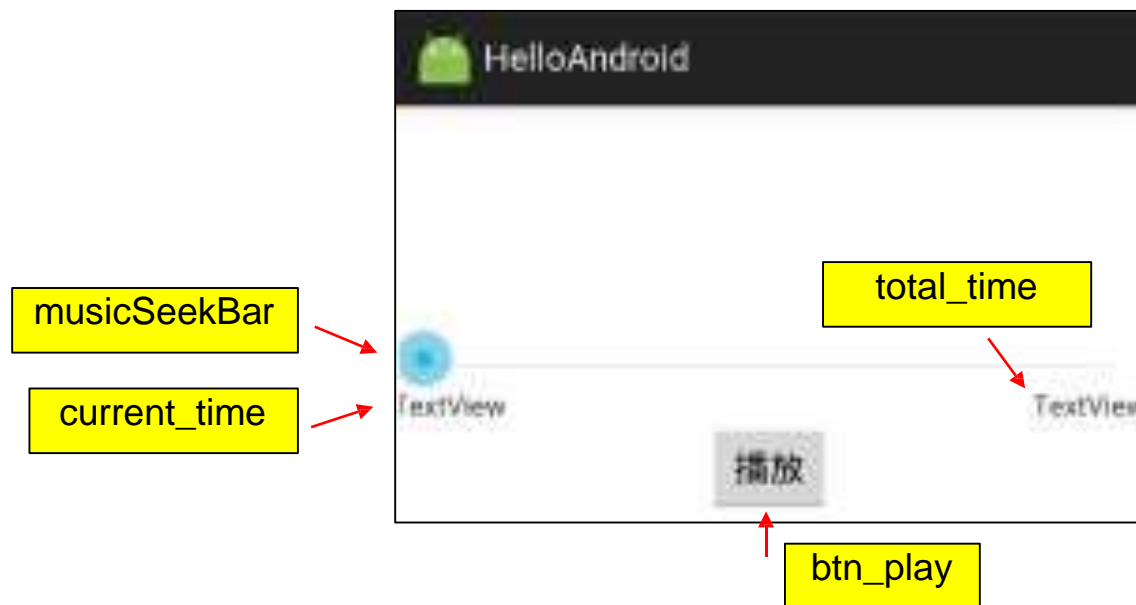


```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
    @Override  
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
        //当拖动条发生变化时调用该方法  
    }  
    @Override  
    public void onStartTrackingTouch(SeekBar seekBar) {  
        //当用户开始滑动滑块时调用该方法  
    }  
    @Override  
    public void onStopTrackingTouch(SeekBar seekBar) {  
        //当用户结束对滑块滑动时,调用该方法  
    }  
});
```

试一下 `tv.setText("progress="+seekBar.getProgress());`

## SeekBar示例 – 音乐播放

- 开启读取权限（读取sdcard中的音乐文件）
- 使用MediaPlayer来播放音乐
- SeekBar关联音乐进度



## 1. 开启读取权限



### ■ AndroidManifest.xml添加权限:

```
<uses-permission  
    android:name="android.permission.READ_EXTERNAL_STORAGE">  
</uses-permission>
```

## 2. 使用MediaPlayer来播放音乐

方法名	功能描述
setDataSource()	设置要播放的音频文件的位置。
prepare()	在开始播放之前调用这个方法完成准备工作。
start()	开始或继续播放音频。
pause()	暂停播放音频。
reset()	将 MediaPlayer 对象重置到刚刚创建的状态。
seekTo()	从指定的位置开始播放音频。
stop()	停止播放音频。调用这个方法后的 MediaPlayer 对象无法再播放音频。
release()	释放掉与 MediaPlayer 对象相关的资源。
isPlaying()	判断当前 MediaPlayer 是否正在播放音频。
getDuration()	获取载入的音频文件的时长。
getCurrentPosition()	获取音乐播放的当前位置

## MediaPlayer结束事件

```
mp.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {  
    @Override  
    public void onCompletion(MediaPlayer mp) {  
        //播放结束  
    }  
});
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_width="match_parent"
```

```
    android:orientation="vertical">
```

```
<SeekBar
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/musicSeekBar"/>
```

```
<LinearLayout
```

```
    android:orientation="horizontal"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">
```

```
<TextView
```

```
    android:text="TextView"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/current_time"
```

```
    android:layout_weight="1"/>
```

```
<TextView
```

```
    android:text="TextView"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/total_time"
```

```
    android:layout_weight="1"
```

```
    android:gravity="right"/>
```

```
</LinearLayout>
```

```
<LinearLayout
```

```
    android:orientation="horizontal"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    android:gravity="center">
```

```
<Button
```

```
    android:text="播放"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/btn_play"/>
```

```
</LinearLayout>
```

```
</LinearLayout>
```



## Activity代码

```
public class MusicActivity extends Activity {  
    MediaPlayer mp=new MediaPlayer(); //音乐播放器  
    SeekBar seekBar;  
    TextView current_time, total_time; //显示当前音乐进度信息  
    Button btn_play; //播放按钮  
  
    int duration=0; //获取时长(ms)  
    int current=0; //当前音乐播放位置  
    boolean isOver=false; //是否播放完毕  
  
    //后续...
```

## Handler定义

```
private mHandler = new Handler() {  
    @Override  
    public void handleMessage(Message msg) {  
        if ( msg.what == 0 ) { //消息标识码为0  
            if( ! isOver ) { //音乐没有结束  
                current = mp.getCurrentPosition();  
                seekBar.setProgress(current);  
                current_time.setText(""+current); //注意""不要掉了  
            }  
        }  
    }  
};  
//后续...
```

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_music);  
  
    //初始化  
    seekBar=(SeekBar)findViewById(R.id.musicSeekBar);  
    current_time=(TextView)findViewById(R.id.current_time);  
    total_time=(TextView)findViewById(R.id.total_time);  
    btn_play=(Button)findViewById(R.id.btn_play);  
  
    //加载音乐  
    String sdpath= Environment.getExternalStorageDirectory().getAbsolutePath();  
    String music=sdpath+"/xiami/audios/漂洋过海来看你_丁当.mp3";  
    try { mp.reset(); //重置  
        mp.setDataSource(music); //加载音乐  
        mp.prepare(); //准备一下  
    } catch (IOException e) { e.printStackTrace(); }  
    // 后续
```

```
duration=mp.getDuration(); //获取音乐时长(ms)
seekBar.setMax(duration); //给SeekBar设置最大时长
```

```
total_time.setText(""+duration);
```

```
btn_play=(Button)findViewById(R.id.btn_play);
```

```
btn_play.setOnClickListener(new View.OnClickListener
```

```
@Override
```

```
public void onClick(View v) {
```

```
    isOver=false;
```

```
    if( mp.isPlaying() ){ mp.pause();
```

```
        btn_play.setText("播放");
```

```
    }else {
```

```
        btn_play.setText("暂停");
```

```
        mp.start(); //播放
```

```
        //启动线程●
```

```
    } //end else
```

```
    } //end onClick
```

```
}); //end setOnClickListener
```

```
//后续...
```

子线程

```
new Thread() {
```

```
    @Override
```

```
    public void run() {
```

```
        while (!isOver) {
```

```
            try {
```

```
                Thread.sleep(50);
```

```
            } catch (InterruptedException e) {
```

```
                e.printStackTrace();
```

```
            }
```

```
            //直接发送一个what=0的空消息
```

```
            mHandler.sendMessage(0);
```

```
        }
```

```
    }
```

```
}.start();
```

**//MediaPlayer播放完毕，复原**

```
mp.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {  
    @Override  
    public void onCompletion(MediaPlayer mp) {  
        isOver=true;  
        btn_play.setText("播放");  
        current=0;  
        current_time.setText(""+current);  
        seekBar.setProgress(0);  
    }  
});  
//后续...
```

//滑动滑块后，音乐从指定位置开始播放

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
    @Override  
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
        //此处无代码  
    }  
    @Override  
    public void onStartTrackingTouch(SeekBar seekBar) {  
        //此处无代码  
    }  
    @Override  
    public void onStopTrackingTouch(SeekBar seekBar) {  
        current=seekBar.getProgress();  
        current_time.setText(""+current);  
        mp.seekTo(current);  
    }  
});  
} //end onCreate
```

```
@Override
```

```
protected void onDestroy() {
```

Activity退出后结束播放

```
    super.onDestroy();
```

```
    isOver=true;
```

← 红色部分很重要

```
    if( mp!=null ){
```

```
        mp.stop();
```

```
        mp.release();
```

```
    }
```

```
    Toast.makeText(getApplicationContext(), "退出啦", Toast.LENGTH_SHORT).show();
```

```
}
```

```
} // end activity (完)
```

## 补充1：用Handler.post()和postDelayed来更新SeekBar进度

```
private Handler handler=new Handler();    //新定义一个Handler
private Runnable update =new Runnable(){    //定义一个Runnable
    @Override
    public void run() {
        if( !isOver ){
            current=mp.getCurrentPosition();
            seekBar.setProgress(current);
            current_time.setText(""+current);
            handler.postDelayed(update, 50);    //每50ms再调用一个Runnable
        }
    }
};
```



```
btn_play=(Button)findViewById(R.id.btn_play);
btn_play.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        isOver=false;
        if(mp.isPlaying()){
            mp.pause();
            btn_play.setText("播放");
        }else {
            btn_play.setText("暂停");
            mp.start();
            handler.post(update); //update这个Runnable需单独定义
        }
    }
});
```

## 补充2：用定时器来更新SeekBar进度

```
btn_play=(Button)findViewById(R.id.btn_play);
btn_play.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        isOver=false;
        if(mp.isPlaying()){
            mp.pause();
            btn_play.setText("播放");
        }else {
            btn_play.setText("暂停");
            mp.start(); //播放
            //定时器代码（见后页）
        }
    }
});
```

① **Timer mTimer = new Timer();** //定义一个计时器

② **TimerTask mTimerTask = new TimerTask() {** //定义任务(也是一个线程)  
 @Override  
 public void **run()** {  
     if( **! isOver** ) {  
         current=mp.getCurrentPosition();  
         seekBar.setProgress(current);  
         **mHandler.post(new Runnable() {** //post线程将在UI主线程中执行  
             @Override  
             public void **run()** {  
                 current\_time.setText(""+current);  
             }  
         }); //end post  
     } //end if  
 } // end run  
 };

//启动定时器, 每50ms执行一次mTimerTask任务,中间0延迟  
 ③ **mTimer.schedule(mTimerTask, 0, 50);**

[【返回】](#)

## 11.5 ListView+SeekBar音乐播放器

- 定义总体布局
- 自定义ListView布局（music\_item.xml）
- 自定义MusicAdpater（Music类和MusicAdpater类）
- 获取手机的所有音乐文件
- ListView、SeekBar相关操作

添加读取外部存储器权限

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

## 定义总体布局



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_width="match_parent"
```

```
    android:orientation="vertical">
```

```
<ListView
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="424dp"
```

```
    android:id="@+id/musicListView"/>
```

```
<SeekBar
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/musicSeekBar"/>
```

```
<LinearLayout
```

```
    android:orientation="horizontal"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">
```

```
<TextView
```

```
    android:text="TextView"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/current_time"
```

```
    android:layout_weight="1"/>
```

```
<TextView
```

```
    android:text="TextView"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/total_time"
```

```
    android:layout_weight="1"
```

```
    android:gravity="right"/>
```

```
</LinearLayout>
```

```
<LinearLayout
```

```
    android:orientation="horizontal"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    android:gravity="center">
```

```
<Button
```

```
    android:text="播放"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/btn_play"/>
```

```
</LinearLayout>
```

```
</LinearLayout>
```

## music\_item.xml布局

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:orientation="horizontal"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    android:background="@drawable/listitem_selector"
```

```
    android:weightSum="1">
```

```
    <TextView
```

```
        android:text="music_id"
```

```
        android:layout_width="0dp"
```

```
        android:layout_height="0dp"
```

```
        android:id="@+id/music_id"
```

```
        android:visibility="invisible"/> ← 隐藏不显示
```

```
    <TextView
```

```
        android:text="music_title"
```

```
        android:layout_width="wrap_content"
```

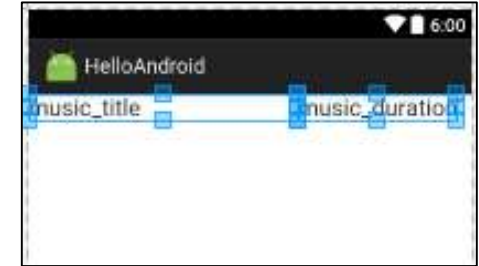
```
        android:layout_height="wrap_content" 音乐名称
```

```
        android:id="@+id/music_title"
```

```
        android:textColor="@drawable/font_selector"
```

```
        android:layout_weight="0.92"
```

```
        android:textSize="20sp"/>
```



```
    <TextView
```

```
        android:text="music_url"
```

```
        android:layout_width="0dp"
```

```
        android:layout_height="0dp"
```

```
        android:id="@+id/music_url"
```

```
        android:visibility="invisible"/> ← 隐藏不显示
```

```
    <TextView
```

```
        android:text="music_duration" 音乐时长
```

```
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
```

```
        android:id="@+id/music_duration"
```

```
        android:textSize="20sp"/>
```

```
</LinearLayout>
```

```
public class Music {  
    int ID; //歌曲ID  
    String Title; //歌曲名称  
    String Url; //歌曲路径  
    int Duration; //歌曲时长  
  
    public Music(int ID, String title, String url, int duration) {  
        this.ID = ID; Title = title; Url = url; Duration = duration;  
    }  
  
    public int getID() { return ID; }  
    public String getTitle() { return Title; }  
    public String getUrl() { return Url; }  
    public int getDuration() { return Duration; }  
    { //set省略 }  
}
```



```
public class MusicAdapter extends ArrayAdapter<Music> {  
    private int resourceId;  
    public MusicAdapter(Context context, int resource, List<Music> objects) {  
        super(context, resource, objects);  
        resourceId=resource;  
    }  
    @Override  
    public View getView(int position, View convertView, ViewGroup parent) {  
        Music m = getItem(position); // 获取当前项的Music实例  
        View view = LayoutInflater.from(getContext()).inflate(resourceId, null);  
        TextView music_id = (TextView) view.findViewById(R.id.music_id);  
        TextView music_title = (TextView) view.findViewById(R.id.music_title);  
        TextView music_url = (TextView) view.findViewById(R.id.music_url);  
        TextView music_duration = (TextView) view.findViewById(R.id.music_duration);  
        music_id.setText(""+m.getID()); //""不能省  
        music_title.setText(m.getTitle());  
        music_url.setText(m.getUrl());  
        music_duration.setText(""+m.getDuration()); //""不能省  
        return view;  
    }  
}
```

将Music对象数据  
逐一绑定到控件  
中

## Activity代码

```
public class MusicActivity extends Activity {  
    ArrayList<Music> musicList = new ArrayList<Music>(); //音乐列表  
    MediaPlayer mp=new MediaPlayer(); //音乐播放器  
    SeekBar seekBar;  
    TextView current_time, total_time; //显示当前音乐进度信息  
    Button btn_play; //播放按钮  
  
    int duration=0; //获取时长(ms)  
    int current=0; //当前音乐播放位置  
    boolean isOver=false; //是否播放完毕  
    //后续
```

## 一些类成员变量

## Handler定义

```
private Handler mHandler = new Handler() {  
    @Override  
    public void handleMessage(Message msg) {  
        if ( msg.what == 0 ) {  
            if( ! isOver ) {  
                current = mp.getCurrentPosition();  
                seekBar.setProgress(current);  
                current_time.setText(""+current);    //""不要掉了  
            }  
        }  
    }  
};
```

//后续

```
public ArrayList<Music> ScannerMusic() {  
    ArrayList<Music> list=new ArrayList<>();  
  
    // 查询媒体数据库  
  
    Cursor cursor = getContentResolver().query(  
        MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, null, null, null,  
        MediaStore.Audio.Media.TITLE);    //按标题排序  
  
    // 遍历媒体数据库  
    if ( cursor.moveToFirst() ){    //遍历每一行  
        while ( !cursor.isAfterLast() ) {    //遍历每一列  
            //歌曲ID: MediaStore.Audio.Media._ID  
            int id = cursor.getInt(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media._ID));  
            //歌曲的名称 : MediaStore.Audio.Media.TITLE  
            String tilte =cursor.getString(  
                cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.TITLE));  
            //后续
```

## 获取手机所有音乐列表(续)

```
//歌曲路径 : MediaStore.Audio.Media.DATA
```

```
String url = cursor.getString(
```

```
        cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.DATA));
```

```
//歌曲时长 : MediaStore.Audio.Media.DURATION
```

```
int duration = cursor.getInt(
```

```
        cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.DURATION));
```

```
Music m=new Music(id,titel,url,duration);
```

```
list.add(m);
```

```
cursor.moveToNext();
```

```
} //end while
```

```
cursor.close();
```

```
} //end if
```

```
return list;
```

```
}
```

```
//后续
```

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_music);  
    //初始化  
    seekBar=(SeekBar)findViewById(R.id.musicSeekBar);  
    current_time=(TextView)findViewById(R.id.current_time);  
    total_time=(TextView)findViewById(R.id.total_time);  
    btn_play=(Button)findViewById(R.id.btn_play);  
    //获取音乐列表  
    musicList=ScannerMusic();  
    //使用MusicAdpater  
    MusicAdapter adapter = new MusicAdapter(  
        MusicActivity.this, R.layout.music_item, musicList );  
    ListView li=(ListView)findViewById(R.id.musicListView);  
    li.setAdapter(adapter);  
    //后续
```

```
li.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        Music m=musicList.get(position);  
        String music_url=m.getUrl(); //获得音乐路径  
        isOver=false;  
        btn_play.setText("暂停");  
        try{  
            mp.reset();    //重置  
            mp.setDataSource(music_url);  
            mp.prepare();    //准备  
        }catch (Exception e){  
            e.printStackTrace();  
        }  
        duration=mp.getDuration(); //获取音乐时长(ms)  
        total_time.setText(""+duration);  
        seekBar.setMax(duration); //给SeekBar设置最大时长 (后续)
```

点击ListView选项事件

```
mp.start(); //播放
```

```
new Thread() {
```

启动子线程

```
    @Override
```

```
    public void run() {
```

```
        while ( ! isOver ) {
```

```
            try {
```

```
                Thread.sleep(50); //每50ms执行一次mTimerTask任务
```

```
            } catch (InterruptedException e) {
```

```
                e.printStackTrace();
```

```
            }
```

```
                mHandler.sendEmptyMessage(0); //直接发送一个what=0的空消息
```

```
            }
```

```
        }
```

```
    }.start();
```

```
    } //end onItemClickListener
```

```
}); //end setOnItemClickListener    后续
```



播放按钮

```
btn_play.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        isOver=false;  
        if( mp.isPlaying() ){  
            mp.pause(); btn_play.setText("播放");  
        }  
        else{  
            mp.start();  
            btn_play.setText("暂停");  
        }  
        //后续，再次启动子线程  
    }  
});
```

```
new Thread() {
```

```
    @Override
```

```
    public void run() {
```

```
        while ( !isOver ) {
```

```
            try {
```

```
                Thread.sleep(50); //每50ms执行一次mTimerTask任务
```

```
            } catch (InterruptedException e) {
```

```
                e.printStackTrace();
```

```
            }
```

```
            mHandler.sendMessage(0); //直接发送一个what=0的空消息
```

```
        }
```

```
    }
```

```
}.start();
```

```
} //end else
```

```
} //end onClick
```

```
}); //end setOnClick
```

```
//后续
```

播放按钮

再次启动子线程(功能同前)

## MediaPlayer播放完毕

//播放完毕

```
mp.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {  
    @Override  
    public void onCompletion(MediaPlayer mp) {  
        isOver=true;  
        btn_play.setText("播放");  
        current=0;  
        current_time.setText(""+current);  
        seekBar.setProgress(0);  
    }  
});  
//后续
```

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
    @Override  
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
        //此处无代码  
    }  
    @Override  
    public void onStartTrackingTouch(SeekBar seekBar) {  
        //此处无代码  
    }  
    @Override  
    public void onStopTrackingTouch(SeekBar seekBar) {  
        current=seekBar.getProgress();  
        current_time.setText(""+current);  
        mp.seekTo(current);  
    }  
}); //end setOnSeekBar  
} // end onCreate
```

SeekBar改变事件

//后续

@Override

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

监测返回键

```
    if( keyCode==event.KEYCODE_BACK ){
```

```
        new AlertDialog.Builder(TestActivity.this)
```

```
            .setIcon(R.drawable.login)
```

```
            .setTitle("提示")
```

```
            .setMessage("确定退出吗? ")
```

```
            .setPositiveButton("确定",new DialogInterface.OnClickListener() {
```

```
                @Override
```

```
                public void onClick(DialogInterface dialog, int which) {
```

```
                    finish();
```

```
                }
```

```
            })
```

```
            .setNegativeButton("取消", null)
```

```
            .show();
```

```
    }
```

```
    return false;
```

```
}
```

// 后续

```
@Override
```

```
protected void onDestroy() {
```

Activity退出后结束播放

```
    super.onDestroy();
```

```
    isOver=true;
```

← 红色部分很重要

```
    if( mp!=null ){
```

```
        mp.stop();
```

```
        mp.release();
```

```
    }
```

```
    Toast.makeText(getApplicationContext(), "退出啦", Toast.LENGTH_SHORT).show();
```

```
}
```

```
} // end activity (完)
```

【完】