实验五：服务与多线程——简单音乐播放器

【实验目的】

1. 学会使用 MediaPlayer；
2. 学会简单的多线程编程，使用 Handle 更新 UI；
3. 学会使用 Service 进行后台工作；
4. 学会使用 Service 与 Activity 进行通信。

【实验内容】

实现一个简单的播放器，要求功能有：

1. 播放、暂停功能；
2. 进度条显示播放进度、拖动进度条改变进度功能；
3. 后台播放功能；
4. 停止功能；
5. 退出功能。

界面仅供参考：

初始化：

播放：

停止后播放：

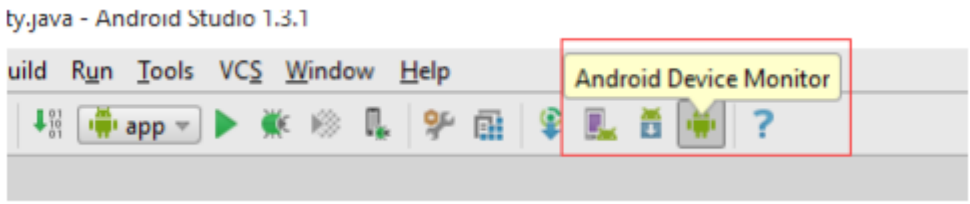
【参考】

1 MediaPlayer

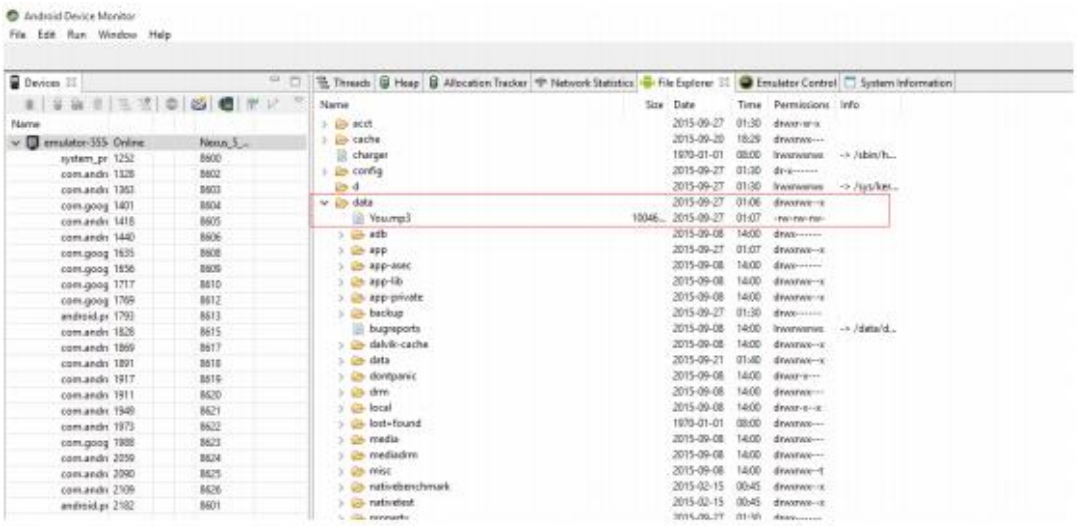
1.1 常用方法：

函数	功能	使用时机
setDataSource(String)	设置音频文件路径 进入初始化状态	MediaPlayer 对象已创建
prepare()	进入就绪状态	已初始化或停止
start()	进入播放状态	已就绪
pause()	进入暂停状态	正在播放
stop()	进入停止状态	正在播放或暂停
isPlaying()	检查是否正在播放	任意正常状态
getCurrentPosition()	获取当前已播放的毫秒数	已就绪
getDuration()	获取文件的时间长度(毫秒)	已就绪
release()	停止播放并释放资源	任何时候

1.2 向虚拟机添加音频文件： 打开 Android Device Monitor， AVD 旁边的就是了：



然后打开 file explorer 选择 data 文件夹点击右上角的导入文件，将音乐文件导入进去



注意不要导入有中文名的文件，要导入有中文名的可以使用 UltraISO 试试。
使用自己手机进行调试的同学， 注意下把文件拷到内置 SD 卡而不是外置 SD 卡会比较方便。
要使用外置的 SD 卡时， 注意下文件路径的获取。 这是相关的路径获取方法:
http://blog.sina.com.cn/s/blog_5da93c8f0102vcam.html
1.3 使用 MediaPlayer：
创建对象：
初始化：
注意下获取的文件路径， 若是使用模拟器的如下， 若是使用自己手机的内置 SD 卡则使用： Environment.getExternalStorageDirectory() + "/data/You.mp3"

```
public static MediaPlayer mp = new MediaPlayer();
public MusicService() {
    try {
        mp.setDataSource("/data/You.mp3");
        mp.prepare();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

播放/暂停：

```
if (mp.isPlaying()) {
    mp.pause();
} else {
    mp.start();
}
```

停止：

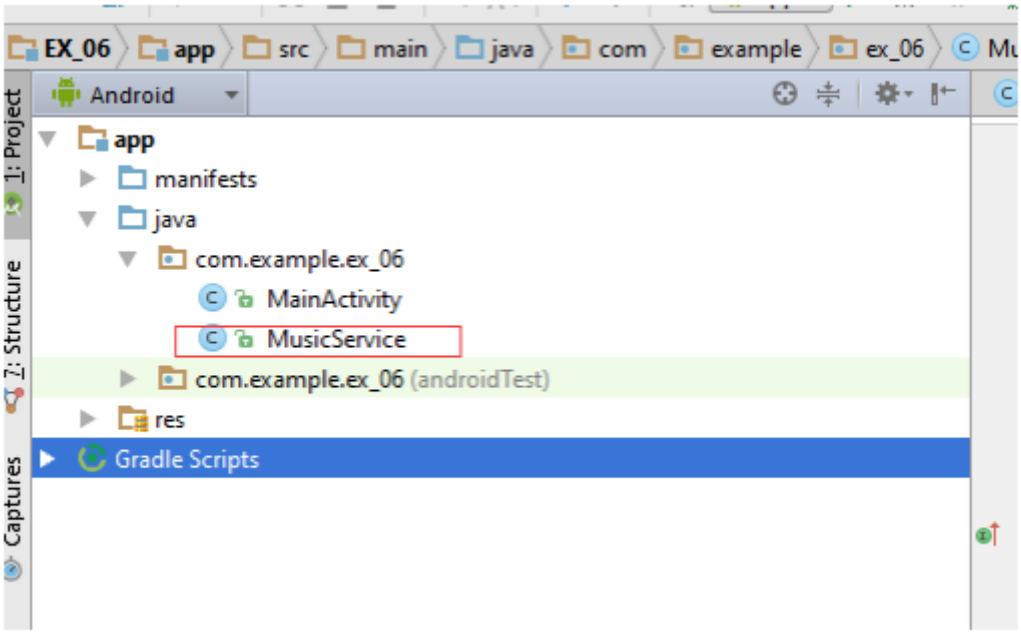
```
public void stop() {
    if (mp != null) {
        mp.stop();
        try {
            mp.prepare();
            mp.seekTo(0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

销毁回收：

```
@Override
public void onDestroy() {
    mp.stop();
    mp.release();
    super.onDestroy();
}
```

2 Service 的使用：

2.1 创建 service 类，实现 MediaPlayer 的功能



2.2 继承 Service 类：

```
public class MusicService extends Service
```

2.3 注册 Service 类：

```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <service android:name=".MusicService" android:exported="true"></service>
</application>

</manifest>
```

2.4 通过 Binder 来保持 Activity 和 Service 的通信(写在 service 类)：

```
*/
public class MusicService extends Service{
    public final IBinder binder = new MyBinder();
    public class MyBinder extends Binder {
        MusicService getService() { return MusicService.this; }
    }
```

2.5 在 Activity 中调用 bindService 保持与 Service 的通信(写在 activity 类)：
Activity 启动时绑定 Service

```
private void bindServiceConnection() {
    Intent intent = new Intent(MainActivity.this, MusicService.class);
//      startService(intent);
    bindService(intent, sc, this.BIND_AUTO_CREATE);
}
```

bindService 成功后回调 onServiceConnected 函数，通过 IBinder 获取 Service 对象，实现 Activity 与 Service 的绑定：

```
};
private ServiceConnection sc = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        musicService = ((MusicService.MyBinder) service).getService();
    }

    @Override
    public void onServiceDisconnected(ComponentName name) { musicService = null; }
};
```

停止服务时，必须解除绑定，写入退出按钮中：

```
handler.removeCallbacks(r);
unbindService(sc);
try {
    //    MainActivity.this.finish();
    System.exit(0);
} catch (Exception e) {

    e.printStackTrace();
}
```

此时，在 Activity 的 onCreate 方法中执行上述与 Service 通信的方法后，即可实现后台播放。点击退出按钮，程序会退出，音乐停止；返回桌面，音乐继续播放。

3 Handler 的使用：

Handler 与 UI 是同一线程，这里可以通过 Handler 更新 UI 上的组件状态， Handler 有很多方法，这里使用比较简便的 post 和 postDelayed 方法。

3.1 使用 Seekbar 显示播放进度：

设置当前值与最大值：

```
seekBar.setProgress(musicService.mp.getCurrentPosition());
seekBar.setMax(musicService.mp.getDuration());
```

3.2 定义 Handler：

run 函数中进行更新 seekbar 的进度在类中定义简单日期格式，用

```
private SimpleDateFormat time = new SimpleDateFormat("m:ss");
```

来显示播放的时间，用 time.format 来格式所需要的数据，用

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
```

来监听进度条的滑动变化：

```
Handler handler = new Handler();
Runnable r = new Runnable() {
    @Override
    public void run() {
        textView.setText(time.format(musicService.mp.getCurrentPosition()) + "/" + time.format(musicService.mp.getDuration()));
        seekBar.setProgress(musicService.mp.getCurrentPosition());
        seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                if (fromUser) {
                    musicService.mp.seekTo(seekBar.getProgress());
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });
        handler.postDelayed(r, 100);
    }
};
```

3.3 按钮实现参考：

播放/暂停：

```
case R.id.btn1:
    musicService.playOrPause();
    textView.setText("ok");
    if (musicService.mp.isPlaying()) {
        state.setText("Playing");
    } else {
        state.setText("Pausing");
    }
    textView.setText(time.format(musicService.mp.getCurrentPosition()) + "/" + time.format(musicService.mp.getDuration()));
    seekBar.setProgress(musicService.mp.getCurrentPosition());
    seekBar.setMax(musicService.mp.getDuration());
    handler.post(r);
    break;
```

停止：

```
case R.id.btn2:
    musicService.stop();
    break;
```

退出：

```
case R.id.btn3:
    handler.removeCallbacks(r);
    unbindService(sc);
    try {
        MainActivity.this.finish();
        System.exit(0);
    } catch (Exception e) {

        e.printStackTrace();
    }
    break;
```

【扩展项】

在原实验基础上，使用 handler 实时更新 UI