课号：	序号：

桂林电子科技大学

Android 应用开发 实验报告

							辅导员意见：
实验名称	实验五 多媒体与 service 实验						
院　　系	计算机与信息安全学院	专业	计算机科学与技术				
学　　号	1900301113	姓名	龚俊源				
实验日期	2022	年	4	月	23	日	成绩　　　辅导员签　名

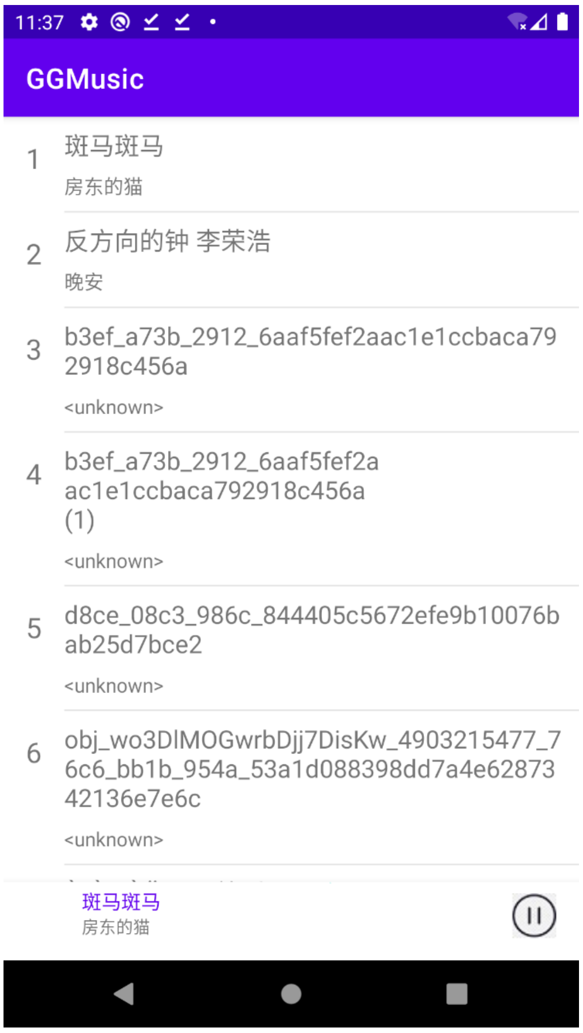
一、 实验目的

1. 掌握 MediaPlayer 进行音频及视频播放的方法；

2. 掌握 MediaPlayer 操作状态图及各状态的含义；

二、 实验内容

程序最终要实现一个音乐播放器，通过实验指导书上的代码进行输入。这次音乐播放器是需要获取外部存储卡的权限，通过更改 AndroidMainFest 的文件进行修改权限设置，一般在手机当中需要获取权限的时候会弹出一个窗口，而这个窗口的实现方法则是在 onCreat 方法当中来获取。获取到了权限之后通过 ContentResolver 类来进行查找外部存储卡中的音乐文件，音乐文件的格式为 mp3。再通过 Adapter 适配器来将查询到的结果绑定到 ListView 这个布局中，再通过 bindView、newView 和 ViewHolder 来显示所查询到的数据。主要的页面布局是通过 ListView 和 activity 两个布局页面来显示，但是音乐播放器当中需要一个底部按钮来控制音乐的播放和暂停，通过 bottom_media_toolbar 的布局来实现。通过 MediaPlayer 方法来实现启动音乐。通过点击下一首播放来实现，则需要在 ListView 中添加 onItemClick 事件来处理点击事件，在 MainActivity 中绑定 MusicServeice 的服务，调用 pause 和 paly 来实现暂停和播放。



1 斑马斑马
房东的猫

2 反方向的钟 李荣浩
晚安

3 b3ef_a73b_2912_6aaf5fef2aac1e1ccbaca792918c456a
<unknown>

4 b3ef_a73b_2912_6aaf5fef2aac1e1ccbaca792918c456a (1)
<unknown>

5 d8ce_08c3_986c_844405c5672efe9b10076bab25d7bce2
<unknown>

6 obj_wo3DlMOGwrbDjj7DisKw_4903215477_76c6_bb1b_954a_53a1d088398dd7a4e6287342136e7e6c
<unknown>

斑马斑马
房东的猫

三、实验总结

在本次实验中主要遇到的问题是配置文件，一开始按照老师指导书上给的版本，没有注意到不同的版本所使用的配置的版本也不一样，就导致 Glide 一直在报红没有得到解决，最终通过在网上查询资料将其版本改成了所用编译器版本和 sdk 所对应的版本，最终能够成功实现。通过实验学会了 MediaPlayer 是怎么样来实现播放音乐，使用 Service 来实现播放和控制音乐。

四、实验代码

1、MainActivity.java

```
package com.example.cord10;

import android.Manifest;
import android.app.job.JobScheduler;
import android.app.job.JobService;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Looper;
import android.os.Message;
import android.provider.MediaStore;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.bumptech.glide.Glide;
import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.io.IOException;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    //
    public    static    final String DATA_URL = "com.example.ggmusic.DATA_URI";
    public    static final String TITLE = "com.example.ggmusic.TITLE";
    public    static final String ARTIST = "com.example.ggmusic.ARTIST";
    public    static final int UPDATE_PROGRESS =1;
    public    static final String ACTION_MUSIC_START =
"com.example.ggmusic.ACTION_MUSIC_START";
    public    static final String ACTION_MUSIC_STOP =
"com.example.ggmusic.ACTION_MUSIC_STOP";
    private MusicReceiver musicReceiver;
    private    MusicService musicService;
    private boolean mBound = false;
    private    boolean mPlayStatus = true;
    private final String TAG = "GGmusic";
    private final int REQUEST_EXTERNAL_STORAGE = 1;
    private ProgressBar progressBar;

    private static String[] PERMISSIONS_STORAGE = {
            Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE
    };
    private ContentResolver mContentResolver;
    private ListView mPlaylist;
    private MediaCursorAdapter mediaCursorAdapter;


    private Cursor mCursor;

    private final String SELECTION = MediaStore.Audio.Media.IS_MUSIC + " = ? " + " AND " +
MediaStore.Audio.Media.MIME_TYPE + " LIKE ? ";

    private final String[] SELECTION_ARGS = {Integer.toString(1), "audio/mpeg"};

    private MediaPlayer mediaPlayer = null;


    private BottomNavigationView navigationView;
    private TextView tvBottomTitle;
    private TextView tvBottomArtist;
    private ImageView ivAlbumThumbnail;
    private    ImageView ivPlay;


    public    class MusicReceiver extends BroadcastReceiver{

                @Override
        public void onReceive(Context context, Intent intent) {
            if(musicService != null)
            {
                if(progressBar ==null)
                {
                    progressBar = findViewById(R.id.progress);
                }
                progressBar.setMax(musicService.getDuration());
                new    Thread(new MusicProgressRunnable()).start();
            }
        }
    }

    private Handler mHandler = new Handler(Looper.getMainLooper())
    {
        public    void handleMessage(Message msg)
        {
            switch (msg.what)
            {
                case UPDATE_PROGRESS:
                    int position = msg.arg1;
                    progressBar.setProgress(position);
                    break;
                default:
                    break;
            }
        }

    };

    private class MusicProgressRunnable implements Runnable
    {
        public    MusicProgressRunnable()
        {

        }

        @Override
        public void run() {
            boolean mThreadWorking = true;
            while (mThreadWorking)
            {
                try {
                    if(musicService != null)
                    {
                        int position = musicService.getCurrentPostion();
                        Message message = new Message();
                        message.what = UPDATE_PROGRESS;
                        message.arg1 = position;
                        mHandler.sendMessage(message);

                    }
                    mThreadWorking = musicService.isPlaying();
```

```
                    Thread.sleep(100);
                }catch (InterruptedException ie)
                {
                    ie.printStackTrace();
                }
            }
        }
    }


    private ListView.OnItemClickListener itemClickListener = new
ListView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

            Log.d(TAG,"点击");
            Cursor cursor = mediaCursorAdapter.getCursor();
            if (cursor != null && cursor.moveToPosition(position)) {
                int titleIndex = cursor.getColumnIndex(MediaStore.Audio.Media.TITLE);
                int artistIndex = cursor.getColumnIndex(MediaStore.Audio.Media.ARTIST);
                int albumIdIndex =
cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM_ID);
                int dataIndex = cursor.getColumnIndex(MediaStore.Audio.Media.DATA);

                String title = cursor.getString(titleIndex);
                String artist = cursor.getString(artistIndex);
                Long albumId = cursor.getLong(albumIdIndex);
                String data = cursor.getString(dataIndex);



                Uri dataUri = Uri.parse(data);

                Intent serviceIntent = new Intent(MainActivity.this,MusicService.class);


                serviceIntent.putExtra(MainActivity.DATA_URL,data);
                serviceIntent.putExtra(MainActivity.TITLE,title);
                serviceIntent.putExtra(MainActivity.ARTIST,artist);
                startService(serviceIntent);


                if(mediaPlayer != null)
                {
                    try {
                        mediaPlayer.reset();
                        mediaPlayer.setDataSource(MainActivity.this,dataUri);
                        //初始化
                        mediaPlayer.prepare();
                        mediaPlayer.start();
                    }catch (IOException ex)
                    {
                        ex.printStackTrace();
                    }

                }
                //更新音乐播放控制栏的信息
                navigationView.setVisibility(View.VISIBLE);
                if(tvBottomTitle!=null)
                {
                    tvBottomTitle.setText(title);
                }
                if(tvBottomArtist != null)
                {
                    tvBottomArtist.setText(artist);
                }

                Uri albumUri =
ContentUris.withAppendedId(MediaStore.Audio.Albums.EXTERNAL_CONTENT_URI,albumId);

                Cursor albumCursor = mContentResolver.query(albumUri,null,null,null,null);

                if(albumCursor != null && albumCursor.getCount()>0)
                {
                    albumCursor.moveToFirst();
                    int albumArtIndex =
albumCursor.getColumnIndex(MediaStore.Audio.Albums.ALBUM_ART);
                    String albumArt = albumCursor.getString(albumArtIndex);
                    Glide.with(MainActivity.this).load(albumArt).into(ivAlbumThumbnail);
                    Log.e(TAG, "onItemClick: "+albumArt );
                    albumCursor.close();
                }
            }
        }
    };

    //连接Service
    private ServiceConnection mConn = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            MusicService.MusicServiceBinder binder =
(MusicService.MusicServiceBinder)service;

            musicService = binder.getService();

            mBound = true;
        }

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

    @Override
    protected void onStart() {
        super.onStart();

        Intent intent = new Intent(MainActivity.this,MusicService.class);

        bindService(intent,mConn, Context.BIND_AUTO_CREATE);

    }

    //释放 mediaplayer 资源

    @Override
    protected void onStop() {

        unbindService(mConn);
        mBound = false;
        super.onStop();
    }

    //获取请求的结果
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
@org.jetbrains.annotations.NotNull String[] permissions, @NonNull
@org.jetbrains.annotations.NotNull int[] grantResults) {
        switch (requestCode)
        {
            case REQUEST_EXTERNAL_STORAGE:
                if(grantResults.length>0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED){ initPlaylist();} break;
            default:break;
        }
```

```
    }

    private void initPlaylist()
    {
        mCursor =
mContentResolver.query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,null,SELECTION,
SELECTION_ARGS,MediaStore.Audio.Media.DEFAULT_SORT_ORDER);
        mediaCursorAdapter.swapCursor(mCursor);
        mediaCursorAdapter.notifyDataSetChanged();
    }



    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mPlaylist = findViewById(R.id.lv_playlist);
        mContentResolver = getContentResolver();
        mediaCursorAdapter = new MediaCursorAdapter(MainActivity.this);
        mPlaylist.setAdapter(mediaCursorAdapter);
        mPlaylist.setOnItemClickListener(itemClickListener);
        progressBar = findViewById(R.id.progress);


if(ContextCompat.checkSelfPermission(this,Manifest.permission.READ_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED) {

if(ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,Manifest.permission.R
EAD_EXTERNAL_STORAGE)){}
            else{

requestPermissions(PERMISSIONS_STORAGE,REQUEST_EXTERNAL_STORAGE);
            }
        }else
        {
            initPlaylist();//初始化播放列表
        }

        navigationView = findViewById(R.id.navigation);


LayoutInflater.from(MainActivity.this).inflate(R.layout.bottom_media_toolbar,navigationView,true);
```

```
        ivPlay = navigationView.findViewById(R.id.iv_play);
        tvBottomArtist = navigationView.findViewById(R.id.tv_bottom_artist);
        tvBottomTitle = navigationView.findViewById(R.id.tv_bottom_title);
        ivAlbumThumbnail = navigationView.findViewById(R.id.iv_thumbnail);

        if(ivPlay != null)
        {
            ivPlay.setOnClickListener(MainActivity.this);

        }
        Log.d(TAG,"test");
        navigationView.setVisibility(View.GONE);

        musicReceiver = new MusicReceiver();
        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(ACTION_MUSIC_START);
        intentFilter.addAction(ACTION_MUSIC_STOP);
        registerReceiver(musicReceiver,intentFilter);
    }

    @Override
    protected void onDestroy() {
        unregisterReceiver(musicReceiver);
        super.onDestroy();
    }


    @Override
    public void onClick(View v) {
        if(v.getId() == R.id.iv_play)
        {
            mPlayStatus =! mPlayStatus;
            if(mPlayStatus == true)
            {
                musicService.play();
                ivPlay.setImageResource(R.drawable.start);
            }
            else
            {
                musicService.pause();
                ivPlay.setImageResource(R.drawable.stop);
            }
        }
    }
}
```

2、MediaCursorAdapter.java

```
package com.example.cord10;


import android.widget.CursorAdapter;
import android.content.Context;
import android.database.Cursor;
import android.provider.MediaStore;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;


class MediaCursorAdapter    extends CursorAdapter {

    private Context mContext;
    private LayoutInflater mLayoutInflater;

    public    MediaCursorAdapter(Context context)
    {
        super(context,null,0);
        mContext = context;
        mLayoutInflater = LayoutInflater.from(mContext);
    }


    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {

        View itemView = mLayoutInflater.inflate(R.layout.list_item,parent,false);

        if(itemView != null)
        {
            ViewHolder vh = new ViewHolder();
            vh.tvTitle = itemView.findViewById(R.id.tv_title);
            vh.tvAritist = itemView.findViewById(R.id.tv_artist);
            vh.tvOrder = itemView.findViewById(R.id.tv_order);
            vh.divider = itemView.findViewById(R.id.divider);
            itemView.setTag(vh);

            return itemView;
        }
        return null;
    }
    /**
     * Bind an existing view to the data pointed to by cursor
     *
     * @param view       Existing view, returned earlier by newView
     * @param context Interface to application's global information
     * @param cursor    The cursor from which to get the data. The cursor is already
     */
    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        ViewHolder vh = (ViewHolder)view.getTag();

        int titleIndex = cursor.getColumnIndex(MediaStore.Audio.Media.TITLE);
        int artistIndex = cursor.getColumnIndex(MediaStore.Audio.Media.ARTIST);

        String title = cursor.getString(titleIndex);
        String artist = cursor.getString(artistIndex);

        int position = cursor.getPosition();

        if(vh!=null)
        {
            vh.tvTitle.setText(title);
            vh.tvAritist.setText(artist);
            vh.tvOrder.setText(Integer.toString(position+1));
        }
    }


    class   ViewHolder{
        TextView tvTitle;
        TextView tvAritist;
        TextView tvOrder;
        View divider;

    }

}

import android.Manifest;
import android.app.AlertDialog;
import android.content.BroadcastReceiver;
import android.content.ContentResolver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
```

```java
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.wang.avi.AVLoadingIndicatorView;

import java.lang.ref.WeakReference;
import java.util.ArrayList;
import java.util.List;

import edu.whut.ruansong.musicplayer.db.MyDbFunctions;
import edu.whut.ruansong.musicplayer.model.ActivityCollector;
import edu.whut.ruansong.musicplayer.model.BaseActivity;
import edu.whut.ruansong.musicplayer.model.DrawerLayoutListViewItem;
import edu.whut.ruansong.musicplayer.model.SongsCollector;
import edu.whut.ruansong.musicplayer.service.MusicService;
import edu.whut.ruansong.musicplayer.R;
import edu.whut.ruansong.musicplayer.model.Song;
import edu.whut.ruansong.musicplayer.tool.DrawerLayoutListViewAdapter;
import edu.whut.ruansong.musicplayer.tool.PictureDealHelper;
import edu.whut.ruansong.musicplayer.tool.SongAdapter;
```

```
public class DisplayActivity extends BaseActivity {
	/*控件*/
	private Toolbar toolbar = null;//toolbar
	private SongAdapter adapter_main_song_list_view;

	private ProgressBar progressBar_activity_display = null;//播放进度条
	private ImageView album_icon = null;//专辑图片
	private TextView play_bar_song_name = null;//歌曲名字
	private TextView play_bar_song_author = null;//歌手
	private ImageButton play_bar_btn_play = null;//底部的图片播放按钮

	private ListView drawer_layout_list_view = null;//侧滑栏的 listView

	private DrawerLayout drawerlayout = null;//侧滑栏

	private MyDbFunctions myDbFunctions;
	private int current_progress = 0;//当前的歌曲播放进度
	private int current_number = 0;//当前正在播放的歌曲
	private int current_status = MusicService.STATUS_STOPPED;//播放状态默认为停止

	private final int REQ_READ_EXTERNAL_STORAGE = 1;//权限请求码,1 代表读取外部存储
权限,2 代表写存储
	private int default_playMode = 0;//默认播放模式,用于打开单选框时默认选中位置的设置
	/*广播接收器*/
	private StatusChangedReceiver statusChangedReceiver = null;//状态接收器，接收来自 service
的播放器状态信息
	private ProgressBarReceiver progressBarReceiver = null;
	/*其它*/

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		/*解决软键盘弹起时，底部控件被顶上去的问题*/

getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_ADJUST_PAN);
		/*设定布局*/
		setContentView(R.layout.activity_display);

		/*启动后台服务*/
		Intent intentService = new Intent(DisplayActivity.this, MusicService.class);
		startService(intentService);

		/*toolbar 相关*/
		initMyToolbar();
		//侧滑菜单界面
		initMyDrawerLayout();
```

```
        /*初始化底部播放控制栏*/
        initControlPlayBar();

        /*获取歌曲数据*/
        load_Songs_data();

        /*启动广播接收器*/
        bindBroadcastReceiver();
    }

    /***************************初始化顶部工具栏*****************/
    private void initMyToolbar(){
        toolbar = findViewById(R.id.toolbar_activity_display);
        drawer_layout_list_view = findViewById(R.id.drawer_layout_list);
        setSupportActionBar(toolbar);
        //更新 toolbar 的标题
        toolbar.setTitle(getResources().getString(R.string.title_toolbar));
    }
    /** toolbar 的 menu 加载*/
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_display, menu);
        return true;
    }

    /* toolbar 的 menu 点击事件*/
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.display_toolbar_menu_search://toolbar 上的搜索按钮
                Intent intent_jump_toolbar_search =
                        new Intent(DisplayActivity.this, SearchDetailActivity.class);
                startActivity(intent_jump_toolbar_search);
                break;
            case R.id.refresh:     //toolbar 上的刷新按钮，用于搜索本地歌曲
                new ScanMusicTask(DisplayActivity.this).execute();
                break;
            default:
                break;
        }
        return true;
    }

    /*侧滑菜单界面*/
    public void initMyDrawerLayout() {
        //drawer_layout 是主界面的最外层布局的 id
        drawerlayout = findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle drawerToggle = new ActionBarDrawerToggle(this, drawerlayout,
toolbar, R.string.app_name, R.string.app_name) {
            @Override
            public void onDrawerOpened(View drawerView) {//完全打开时触发
                super.onDrawerOpened(drawerView);

//Toast.makeText(DisplayActivity.this,"onDrawerOpened",Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onDrawerClosed(View drawerView) {//完全关闭时触发
                super.onDrawerClosed(drawerView);

//Toast.makeText(DisplayActivity.this,"onDrawerClosed",Toast.LENGTH_SHORT).show();
            }

            /**
             * 当抽屉被滑动的时候调用此方法
             * slideOffset 表示 滑动的幅度（0-1）
             */
            @Override
            public void onDrawerSlide(View drawerView, float slideOffset) {
                super.onDrawerSlide(drawerView, slideOffset);
            }

            /**
             * 当抽屉滑动状态改变的时候被调用
             * 状态值是 STATE_IDLE（闲置--0）, STATE_DRAGGING（拖拽的--1）,
STATE_SETTLING（固定--2）中之一。
             *具体状态可以慢慢调试
             */
            @Override
            public void onDrawerStateChanged(int newState) {
                super.onDrawerStateChanged(newState);
            }
        };
        drawerlayout.setDrawerListener(drawerToggle);
        //设置 toolbar 左侧图标点击打开侧滑栏
        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (drawerlayout.isDrawerOpen(GravityCompat.START)) {
                    //Log.w("DisplayActivity", "closeDrawer");
                    drawerlayout.closeDrawer(GravityCompat.START);
                } else {
                    //Log.w("DisplayActivity", "openDrawer");
                    drawerlayout.openDrawer(GravityCompat.START);
                }
            }
        });

        //配置侧滑界面 listView
        List<DrawerLayoutListViewItem> drawer_list_view_content = new ArrayList<>();
        DrawerLayoutListViewItem play_mode_select = new
DrawerLayoutListViewItem(R.drawable.setting, "播放模式");
        DrawerLayoutListViewItem exit = new DrawerLayoutListViewItem(R.drawable.exit, "退
出");
        drawer_list_view_content.add(play_mode_select);
        drawer_list_view_content.add(exit);
        DrawerLayoutListViewAdapter drawer_list_view_adapter = new
DrawerLayoutListViewAdapter(DisplayActivity.this, R.layout.drawer_layout_list_item,
drawer_list_view_content);
        drawer_layout_list_view.setAdapter(drawer_list_view_adapter);
        /*设置侧滑栏 listView 的 item 点击事件*/
        drawer_layout_list_view.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int position, long
id) {
                switch (position) {
                    case 0:
                        //播放模式选择
                        if (SongsCollector.size() != 0) {
                            selectMode();
                            drawerlayout.closeDrawer(GravityCompat.START);
                        } else {
                            Toast.makeText(DisplayActivity.this, "本机无歌曲,请下载！",
Toast.LENGTH_SHORT).show();
                        }
                        break;
                    case 1://退出
                        Intent stop_service_intent = new Intent(DisplayActivity.this,
MusicService.class);
                        stopService(stop_service_intent);
                        ActivityCollector.finishAll();
                        System.exit(0);
                        break;
                    default:
                        break;
                }
            }
        });
    }

    /*选择播放模式*/
    public void selectMode() {
        final AlertDialog.Builder builder = new AlertDialog.Builder(DisplayActivity.this);
        builder.setIcon(R.drawable.setting);
        builder.setTitle("播放模式");
        final String[] mode = {"顺序播放", "单曲循环", "随机播放"};
        /*设置一个单项选择框
         * 第一个参数指定要显示的一组下拉单选框的数据集合
         * 第二个参数代表索引，指定默认哪一个单选框被勾选上，0 表示默认'顺序播放'
会被勾选上
         * 第三个参数给每一个单选项绑定一个监听器
         */
        final Intent intent_mode = new
Intent(MusicService.BROADCAST_MUSICSERVICE_CONTROL);
        builder.setSingleChoiceItems(mode, default_playMode, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(DisplayActivity.this,
                        "播放模式为：" + mode[which], Toast.LENGTH_SHORT).show();
                switch (which) {
                    case 0:
                        intent_mode.putExtra("command",
MusicService.PLAY_MODE_ORDER);
                        dialog.cancel();
                        break;
                    case 1:
                        intent_mode.putExtra("command",
MusicService.PLAY_MODE_LOOP);
                        default_playMode = 1;
                        dialog.cancel();
                        break;
                    case 2:
                        intent_mode.putExtra("command",
MusicService.PLAY_MODE_RANDOM);
                        default_playMode = 2;
                        dialog.cancel();
                        break;
                    default:
                        break;
                }
                sendBroadcast(intent_mode);
            }
        });
        builder.show();
    }

    /*--------------顶部工具栏相关结束----------------------------*/



    /*----------------------------初始化歌曲列表控件-------------*/
    private void initMySongListView(){
        //歌曲列表
        final ListView main_song_list_view = findViewById(R.id.main_song_list_view);
        adapter_main_song_list_view = new SongAdapter(DisplayActivity.this,
                R.layout.song_list_item, SongsCollector.getSongsList());
        main_song_list_view.setAdapter(adapter_main_song_list_view);
        /*设置歌曲列表 item 点击事件*/
        main_song_list_view.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                                    int position, long id) {
                if (current_status == MusicService.STATUS_PLAYING) {//播放状态
                    if (current_number == position) {//点击的正在播放的歌曲
                        Log.w("DisplayActivity","点击的正在播放的歌曲");
                        sendBroadcastOnCommand(MusicService.COMMAND_PAUSE);//暂
停
                    } else {//点击的别的歌曲
                        current_number = position;
                        sendBroadcastOnCommand(MusicService.COMMAND_PLAY);
                    }
                } else if (current_status == MusicService.STATUS_PAUSED) {//暂停状态
                    if (current_number == position) {
                        //应恢复播放
                        sendBroadcastOnCommand(MusicService.COMMAND_RESUME);
                    } else {
                        //点击的别的歌曲
                        current_number = position;
                        sendBroadcastOnCommand(MusicService.COMMAND_PLAY);
                    }
                } else {//停止状态
                    current_number = position;
                    sendBroadcastOnCommand(MusicService.COMMAND_PLAY);
                }
            }
        });
    }
    /*--------------------歌曲列表控件初始化结束--------------------*/


    /*--------------------初始化底部播放控制栏--------------------*/
    private void initControlPlayBar(){
        progressBar_activity_display = findViewById(R.id.progressBar_activity_display);
        //底部播放控制栏
        View play_bar_bottom = findViewById(R.id.play_bar_bottom);
        /*点击底部一栏的事件*/
        play_bar_bottom.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {//跳转到歌曲详情页
                if (SongsCollector.size() != 0) {
                    Intent intent = new Intent(DisplayActivity.this, SongDetailActivity.class);
                    intent.putExtra("current_number", current_number);
                    intent.putExtra("current_status", current_status);
                    intent.putExtra("current_progress", current_progress);
                    intent.putExtra("current_PlayMode", default_playMode);
                    startActivity(intent);
                } else
                    Toast.makeText(DisplayActivity.this, "别点啦,不会有结果的",
Toast.LENGTH_SHORT).show();
            }
        });

        album_icon = findViewById(R.id.album_icon);
        play_bar_song_name = findViewById(R.id.play_bar_song_name);
        play_bar_song_author = findViewById(R.id.play_bar_song_author);

        play_bar_btn_play = findViewById(R.id.play_bar_btn_play);
        /*初始化播放按钮点击事件*/
        play_bar_btn_play.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (SongsCollector.size() != 0) {
                    switch (current_status) {
                        case MusicService.STATUS_PLAYING:
                            Log.w("DisplayActivity","歌曲正在播放,点击了 play 按钮,所以
暂停");
                            sendBroadcastOnCommand(MusicService.COMMAND_PAUSE);
                            break;
                        case MusicService.STATUS_PAUSED:
                            //Log.w("DisplayActivity", "STATUS_PAUSED");

sendBroadcastOnCommand(MusicService.COMMAND_RESUME);
```

```
                                break;
                            case MusicService.STATUS_STOPPED:
                                //Log.w("DisplayActivity", "STATUS_STOPPED");
                                sendBroadcastOnCommand(MusicService.COMMAND_PLAY);
                                break;
                            default:
                                break;
                        }
                    } else {
                        Toast.makeText(DisplayActivity.this, "本机无歌曲,请下载！ ",
Toast.LENGTH_SHORT).show();
                    }
                }
            });

            //下一首歌曲按钮
            ImageButton play_bar_btn_next = findViewById(R.id.play_bar_btn_next);
            /*初始化下一首按钮点击事件*/
            play_bar_btn_next.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    if (SongsCollector.size() != 0) {
                        sendBroadcastOnCommand(MusicService.COMMAND_NEXT);
                    } else {
                        Toast.makeText(DisplayActivity.this, "本机无歌曲,请下载！ ",
Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }

        /*设置底部的一栏左侧的歌曲名和歌手以及专辑图片*/
        private void updateBottomMes(int position) {//
                Song song = SongsCollector.getSong(position);//获取点击位置的 song 对象
            play_bar_song_name.setText(song.getTitle());
            play_bar_song_author.setText(song.getArtist());
            //设置专辑图片
            album_icon.setImageBitmap(PictureDealHelper.getAlbumPicture(this,
SongsCollector.getSong(current_number).getDataPath(), 120, 120));
        }


        /*---------------------底部播放控制栏初始化结束---------------------*/

        /*---------------------存储权限以及歌曲数据---------------------*/
```

```
    /*加载歌曲数据*/
    private void load_Songs_data() {
        if (SongsCollector.size() == 0) {
            myDbFunctions = MyDbFunctions.getInstance(this);//获取数据库操作类实例
            if (!myDbFunctions.isSONGS_Null()) {
                //数据库里面有数据,直接加载数据库里面的
                new GetDB_DataTask(DisplayActivity.this).execute();
                return;
            }
            /*判断是否需要请求权限,然后获取歌曲数据*/
            requestPermissionByHand();
        }else{
            initMySongListView();//已有歌曲,直接初始化控件
        }
    }
    /*向用户请求权限*/
    private void requestPermissionByHand() {
        //判断当前系统的版本
        if (Build.VERSION.SDK_INT >= 23) {//6.0 以上
            int checkReadStoragePermission = ContextCompat.checkSelfPermission(
                        DisplayActivity.this,
Manifest.permission.READ_EXTERNAL_STORAGE);
            //如果读取没有被授予
            if (checkReadStoragePermission != PackageManager.PERMISSION_GRANTED) {
                //请求权限,此处可以同时申请多个权限
                ActivityCompat.requestPermissions(
                        DisplayActivity.this, new String[]{
                                Manifest.permission.READ_EXTERNAL_STORAGE
                        }, REQ_READ_EXTERNAL_STORAGE);
            } else {//已有权限,加载歌曲
                new ScanMusicTask(DisplayActivity.this).execute();
            }
        }
        else{
            new ScanMusicTask(DisplayActivity.this).execute();
        }
    }

    /*向用户请求权限后的回调*/
    @Override
    public void onRequestPermissionsResult(int requestCode, final String[] permissions, int[]
grantResults) {
        if (requestCode == REQ_READ_EXTERNAL_STORAGE) {
            //  如果请求被取消了，那么结果数组就是空的
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
```

```
                    // 权限被授予了
                    if (SongsCollector.size() == 0) {
                        new ScanMusicTask(DisplayActivity.this).execute();
//                        scanMusic();//加载歌曲数据
                    }
                } else {
                    Toast.makeText(DisplayActivity.this, "读存储权限申请失败",
Toast.LENGTH_SHORT).show();
                }
            }
        }

    /*扫描歌曲文件*/
    /*
     * Params：开始异步任务执行时传入的参数类型；
     * Progress：异步任务执行过程中，返回下载进度值的类型；
     * Result：异步任务执行完成后，返回的结果类型
     *  静态内部类+弱引用防止内存泄漏
     * */
    private static class GetDB_DataTask extends AsyncTask<Void, Void, Boolean> {
        private final WeakReference<DisplayActivity> displayActivityWeakReference ;
        GetDB_DataTask(DisplayActivity displayActivity){
            displayActivityWeakReference = new WeakReference<>(displayActivity);
        }
        //在后台任务开始执行之间调用，在主线程执行
        @Override
        protected void onPreExecute() {
            DisplayActivity displayActivity = displayActivityWeakReference.get();
            AVLoadingIndicatorView loading_animation =
displayActivity.findViewById(R.id.loading_animation);
            loading_animation.setVisibility(View.VISIBLE);//显示加载动画
        }

        //在子线程中运行，处理耗时任务
        @Override
        protected Boolean doInBackground(Void... params) {
            DisplayActivity displayActivity = displayActivityWeakReference.get();
            if(displayActivity != null){
                if(displayActivity.myDbFunctions != null){
                    if (!displayActivity.myDbFunctions.isSONGS_Null()) {
                        //数据库里面有数据,直接加载数据库里面的

SongsCollector.setSongsList(displayActivity.myDbFunctions.loadAllSongs());
                        SongsCollector.song_total_number = SongsCollector.size();
                        return true;
                    }
                    }
                }
                return false;
            }
            //返回的数据会作为参数传递到此方法中，可以利用返回的数据来进行一些 UI 操
作，在主线程中进行
            @Override
            protected void onPostExecute(Boolean result) {
                DisplayActivity displayActivity = displayActivityWeakReference.get();
                if (result) {
                    /*初始化歌曲列表控件,必须在获取数据后面*/
                    displayActivity.initMySongListView();
                } else {
                    Toast.makeText(displayActivity, "加载歌曲数据失败",
Toast.LENGTH_SHORT).show();
                }
                AVLoadingIndicatorView loading_animation =
displayActivity.findViewById(R.id.loading_animation);
                loading_animation.setVisibility(View.GONE);//加载动画消失
            }
        }

        private static class ScanMusicTask extends AsyncTask<Void, Void, Boolean> {
            private final WeakReference<DisplayActivity> displayActivityWeakReference ;
            ScanMusicTask(DisplayActivity displayActivity){
                displayActivityWeakReference = new WeakReference<>(displayActivity);
            }

            @Override
            protected void onPreExecute() {
                DisplayActivity displayActivity = displayActivityWeakReference.get();
                AVLoadingIndicatorView loading_animation =
displayActivity.findViewById(R.id.loading_animation);
                loading_animation.setVisibility(View.VISIBLE);//加载动画显示
            }


            @Override
            protected Boolean doInBackground(Void... voids) {
                DisplayActivity displayActivity = displayActivityWeakReference.get();
                ContentResolver contentResolver = displayActivity.getContentResolver();
                try (Cursor cursor =
contentResolver.query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
                        null, null, null, null)) {
                    if (cursor != null) {
                        while (cursor.moveToNext()) {
```

```
                                        //是否是音频
                                        int isMusic =
cursor.getInt(cursor.getColumnIndex(MediaStore.Audio.Media.IS_MUSIC));
                                        //时长
                                        long duration =
cursor.getLong(cursor.getColumnIndex(MediaStore.Audio.Media.DURATION));
                                        //是音乐并且时长大于 2 分钟
                                        if (isMusic != 0 && duration >= 2 * 60 * 1000) {
                                             //文件路径
                                             String dataPath =
cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.DATA));
                                             if (SongsCollector.isContainSong(dataPath)) {//数据库中已经有
这首歌曲了,所以跳过
                                                  continue;
                                             }
                                             //歌名
                                             String title =
cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.TITLE));
                                             //歌手
                                             String artist =
cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.ARTIST));
                                             //专辑 id
                                             //歌名，歌手，时长,文件路径,是否喜爱,专辑,专辑图片,是否使
用默认专辑图片
                                             Song song = new Song(
                                                       title,
                                                       artist,
                                                       duration,
                                                       dataPath,
                                                       PictureDealHelper.getAlbumPicture(dataPath,96,96),
                                                       false
                                             );//R.drawable.song_item_picture 是歌曲列表每一项前面那个图
标
                                             if(!SongsCollector.isContainSong(song.getDataPath())){//只添加
当前列表中没有
                                                  SongsCollector.addSong(song);
                                                  if(song.getAlbum_icon() == null){
                                                       song.setFlagDefaultAlbumIcon(true);
                                                  }

displayActivityWeakReference.get().myDbFunctions.saveSong(song);
                                             }//只添加当前列表中没有
                                        }//是音乐并且时长大于 2 分钟
                                   }//游标的移动
                              }//游标不为空
                         } catch (Exception e) {
```

```
                    e.printStackTrace();
                }
                return true;
            }

            //返回的数据会作为参数传递到此方法中，可以利用返回的数据来进行一些 UI 操
作，在主线程中进行
            @Override
            protected void onPostExecute(Boolean result) {
                DisplayActivity displayActivity = displayActivityWeakReference.get();
                if(result){
                    Toast.makeText(displayActivity,"歌曲扫描完毕
",Toast.LENGTH_SHORT).show();
                    //给一些歌曲添加默认专辑图片
                    for(int i = 0; i < SongsCollector.size(); i ++){
                        Song s = SongsCollector.getSong(i);
                        if(s.getAlbum_icon() == null){
                            Bitmap b = PictureDealHelper.getAlbumPicture(displayActivity,
                                    s.getDataPath(),96,96);
                            s.setAlbum_icon(b);
                        }
                    }
                    /*初始化歌曲列表控件,必须在获取数据后面*/
                    displayActivity.initMySongListView();
                    AVLoadingIndicatorView loading_animation =
displayActivity.findViewById(R.id.loading_animation);
                    loading_animation.setVisibility(View.GONE);//加载动画消失
                    if (SongsCollector.size() == 0) {
                        Toast.makeText(displayActivity, "本机无歌曲,请下载！",
Toast.LENGTH_SHORT).show();
                    }
                }else{
                    Toast.makeText(displayActivity,"默认专辑图片出错
",Toast.LENGTH_SHORT).show();
                }
            }
        }

        /*--------------------存储权限以及歌曲数据---结束--------------------*/

        /**
         *  由不可见变为可见的时候调用
         */
        @Override
        protected void onStart() {
            super.onStart();
            //广播接收器重新注册
            if (progressBarReceiver == null) {
                bindBroadcastReceiver();
            }
            //重新加载底部栏的歌名,歌手,专辑图片,播放按钮 UI
            current_number = MusicService.getCurrent_number();
            current_status = MusicService.getCurrent_status();
            if (SongsCollector.size() != 0) {//避免空引用错误
                play_bar_song_name.setText(SongsCollector.getSong(current_number).getTitle());
                play_bar_song_author.setText(SongsCollector.getSong(current_number).getArtist());
                album_icon.setImageBitmap(
                        PictureDealHelper.getAlbumPicture(this,
SongsCollector.getSong(current_number).getDataPath(), 120, 120));
                if (current_status == MusicService.STATUS_PLAYING) {
                    //正在播放
                    play_bar_btn_play.setBackground(getDrawable(R.drawable.pause_32));
                } else {
                    play_bar_btn_play.setBackground(getDrawable(R.drawable.play_32));
                }
            }
        }

        /**
         * 准备好和用户进行交互的时候调用
         */
        @Override
        protected void onResume() {
            super.onResume();
            sendBroadcastOnCommand(MusicService.COMMAND_REQUEST_DURATION);
        }

        /**
         * Activity 正在停止，仍可见
         */
        @Override
        protected void onPause() {
            super.onPause();
        }

        /**
         * Activity 即将停止，不可见，位于后台,可以做稍微重量级的回收工作
         */
        @Override
        protected void onStop() {
            super.onStop();
        }
```

```
/**
  * Activity 即将销毁,做一些最终的资源回收
  */
@Override
protected void onDestroy() {
      super.onDestroy();
      //取消广播接收器的注册
      if (statusChangedReceiver != null)
            unregisterReceiver(statusChangedReceiver);
      if (progressBarReceiver != null)
            unregisterReceiver(progressBarReceiver);
      if (current_status == MusicService.STATUS_STOPPED) {
            stopService(new Intent(this, MusicService.class));
      }
}

/**
  *  回退键      不返回登录界面
  */
@Override
public void onBackPressed() {
      super.onBackPressed();
      ActivityCollector.finishAll();
}

/*******绑定广播接收器,接收来自服务的广播*/
private void bindBroadcastReceiver() {
      //播放器状态接收
      statusChangedReceiver = new StatusChangedReceiver();
      IntentFilter intentFilter = new
IntentFilter(MusicService.BROADCAST_MUSICSERVICE_UPDATE_STATUS);
      registerReceiver(statusChangedReceiver, intentFilter);
      //进度条相关广播
      progressBarReceiver = new ProgressBarReceiver();
      IntentFilter intentFilter1 = new
IntentFilter(MusicService.BROADCAST_MUSICSERVICE_PROGRESS);
      registerReceiver(progressBarReceiver, intentFilter1);
}

/***发送命令，控制音乐播放，参数定义在 MusicService 中*/
private void sendBroadcastOnCommand(int command) {
      //1.创建 intent,控制命令
      Intent intent = new Intent(MusicService.BROADCAST_MUSICSERVICE_CONTROL);
      //2.封装数据
      intent.putExtra("command", command);
```

```
        switch (command) {
            case MusicService.COMMAND_PLAY:
                intent.putExtra("number", current_number);//封装歌曲在 list 中的位置
                break;
            case MusicService.COMMAND_RESUME:
            case MusicService.COMMAND_PAUSE:
            case MusicService.COMMAND_REQUEST_DURATION:
            default:
                break;
        }
        //3.发送广播
        sendBroadcast(intent);
    }

    /*****内部类，接受播放器状态更改广播命令并执行操作*/
    class StatusChangedReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
            //获取播放器状态
            int status = intent.getIntExtra("status", -1);
            if (status != MusicService.PLAY_MODE_UPDATE)
                current_status = status;
            switch (status) {
                //播放器状态更改为正在播放
                case MusicService.STATUS_PLAYING:
                    //把底部播放按钮的图标改变,列表中正在播放的歌曲的颜色改变
                    Log.w("DisplayActivity", "STATUS_PLAYING");
                    play_bar_btn_play.setBackground(getDrawable(R.drawable.pause_32));//改
变图标
                    current_number = MusicService.getCurrent_number();//更改存储的当前播
放歌曲序号
                    //加载歌名和歌手,设置专辑图片
                    updateBottomMes(current_number);
                    //通知适配器数据变化
                    adapter_main_song_list_view.notifyDataSetChanged();
                    break;

                //播放器状态更改为暂停
                case MusicService.STATUS_PAUSED:
                    Log.w("DisplayActivity", "STATUS_PAUSED");
                    play_bar_btn_play.setBackground(getDrawable(R.drawable.play_32));//把
底部播放按钮的图标改变
                    //通知适配器数据变化
                    adapter_main_song_list_view.notifyDataSetChanged();
                    break;
                //音乐播放服务已停止
                case MusicService.STATUS_STOPPED:
                    Log.w("DisplayActivity", "STATUS_STOPPED");
                    ActivityCollector.finishAll();
                    break;

                //播放器状态更改为播放完成
                case MusicService.STATUS_COMPLETED:
                    Log.w("DisplayActivity", "STATUS_COMPLETED");
                    break;
                case MusicService.PLAY_MODE_UPDATE:
                    //顺序,单曲,随机  --->   8,9,10
                    //在弹窗中位置分别是 0,1,2
                    default_playMode = intent.getIntExtra("playMode",
MusicService.PLAY_MODE_ORDER) - 8;
                    break;
                default:
                    break;
            }
        }
    }

    /**
     * 内部类，接受 service 广播动态更新 progressBar
     */
    class ProgressBarReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
            //进度条的广播命令
            int progress_broadcast_content = intent.getIntExtra("content", 0);
            switch (progress_broadcast_content) {
                case MusicService.PROGRESS_DURATION:
                    /*用于存储*/
                    //当前的歌曲的总时长
                    int duration = intent.getIntExtra("duration", 0);
                    progressBar_activity_display.setMax(duration);
                    break;
                case MusicService.PROGRESS_UPDATE:
                    current_progress = intent.getIntExtra("current_progress", 0);
                    progressBar_activity_display.setProgress(current_progress);
                    break;
                default:
                    break;
            }
        }
    }
```

```
	@Override
	public boolean onKeyDown(int keyCode, KeyEvent event) {
		if (KeyEvent.KEYCODE_HEADSETHOOK == keyCode) { //按下了耳机键
			switch (current_status) {
				case MusicService.STATUS_PLAYING:
					Log.w("DisplayActivity", "按下了耳机键");
					sendBroadcastOnCommand(MusicService.COMMAND_PAUSE);
					break;
				case MusicService.STATUS_PAUSED:
					sendBroadcastOnCommand(MusicService.COMMAND_RESUME);
					break;
				case MusicService.STATUS_STOPPED:
					sendBroadcastOnCommand(MusicService.COMMAND_PLAY);
					break;
				default:
					break;
			}
		}
		return super.onKeyDown(keyCode, event);
	}
}
```