软件说明文档

软件名称：音乐播放器

完成人：朱旭

学号：20152100134

完成日期：2017-12-15

一、软件内容简介

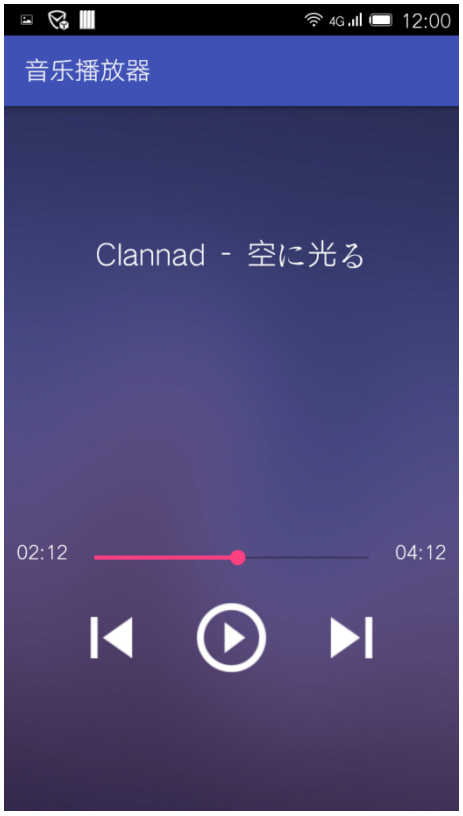
本软件基于 Service 实现音乐播放器。

该音乐播放器获取读取手机存储的权限，获取本地音乐列表。

点击音乐进入音乐播放界面，同时启动 Service 播放音乐。对音乐可以进行播放、暂停、上一首、下一首等操作。进度条监听和显示播放进度，滑动进度条也可以改变播放时间。音乐的播放模式默认为自动播放和列表循环。

二、界面设计

该音乐播放器的主界面基于 ListView 加载本地音乐文件，点击歌曲通过 Intent 跳转到播放界面（播放界面采用相对布局），同时启动 Service 播放音乐。



三、软件操作流程

主界面：

点击歌曲进入播放界面：

滑动进度条：

暂停（暂停播放，按钮图标改变）：

点击下一首（上一首类似）：

四、难点与解决方法

1、多线程问题

【难点】在子线程中更新 UI 会导致界面闪退

【解决办法】Android 程序启动时，会开启程序的主线程，也叫 UI 线程，UI 线程控制着 UI 界面中的控件，并进行事件的分发，更新 UI 等操作只能在 UI 线程当中完成。如果在耗时的子线程当中更新 UI，会导致程序崩溃闪退。例如下面的代码：

```
Timer timer = new Timer();
timer.schedule(new TimerTask() {
@Override
public void run() {
    tv_name.setText("text");
}
}, 0, 1000);
```

在定时器的子线程中更新 UI 导致程序崩溃。

为了解决上面的问题，可以采用 Handler 进行数据的异步回调。在 Android 中，Handler 负责消息的传递和进行 UI 线程的更新操作。形象地说，Handler 是主线程和子线程沟通的桥梁。

一般情况下，主线程绑定了 Handler 对象，并在事件触发上面创建子线程用于某些耗时操作，当子线程的工作完成后，会向 Handler 发送一个已完成的信号（Message 对象），当 Handler 接收到信号后，就会对主线程 UI 进行更新操作（课本 P206 Handler 消息机制）。

Handler 向主线程发送消息更新 UI 可以通过以下方法：

post(Runnable),

postAtTime(Runnable, long),

postDelayed(Runnable,long),

sendEmptyMessage(int),

sendMessage(Message),

sendMessageAtTime(Message,long),

sendMessageDelayed(Message, long)

对于之前的程序闪退的代码，我们可以进行如下修改：

```
final Handler handler = new Handler();
Timer timer = new Timer();
timer.schedule(new TimerTask() {
    @Override
    public void run() {
        handler.post(new Runnable() {
            @Override
            public void run() {
                tv_name.setText("text");
            }
```

```
            });
        }
    }, 0, 1000);
```

先定义一个 Handler 对象，然后在子线程中调用该对象的 post(Runable)方法，向主线程发送消息，进行 UI 的更新。

2、程序中的异步问题

【难点】该程序中某些异步操作不当会导致程序闪退。

【解决办法】该程序中涉及的异步操作有：动态更新 UI、绑定服务、MediaPlayer 加载（prepareAsync()）等。

（1）动态更新 UI：在某些耗时操作中更新 UI，可以通过 Handler 进行消息传递。

（2）绑定服务：绑定是异步的，bindService()会立即返回，它不会返回 IBinder 给客户端。要接收 IBinder，客户端必须创建一个 ServiceConnection 的实例并传 bindService()。ServiceConnection 包含一个回调方法，系统调用这个方法来传递要返回的 IBinder。

看如下的代码：

```
bindService(intent_music, conn, BIND_AUTO_CREATE);
binder.initialPlayer();
```

上面的代码想实现在绑定服务后的调用 IBinder 对象的某个方法，但是因为绑定服务是异步的，在 IBinder 对象返回之前，我们就对该对象进行引用，就会因为空指针引用等问题导致程序闪退。解决的办法是在 ServiceConnection 的实例的回调方法 onServiceConnected()中才对 IBinder 对象进行调用：

```
private class MyConn implements ServiceConnection {

    @Override
    public void onServiceConnected(ComponentName name, IBinder
service){
        binder = (MusicService.MyBinder) service;
        binder.initialPlayer();
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {

    }
}
```

（2）MediaPlayer 异步加载问题：MediaPlayer 的加载可以分为同步加载和异步加载，其中 Mediaplayer 的 prepare()进行同步加载，MediaPlayer.create()创建对象时也进行了同步加载；而 prepareAsync()进行的是异步加载，如果在 MediaPlayer 对象加载完媒体文件之前调用对象，也会引起程序的闪退问题，解决办法是将对象的调用写在对象的监听函数 setOnDrmPreparedListener()中。

五、不足之处及今后感想

1、访问权限。在本次开发中，进行测试的手机的版本为 Android 4.4.4，Android6.0 以下的设备设置访问本地文件的权限只需要在 Manifests 文件中添加<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" /> 即可，在 Android6.0 及以上版本中，需要设置运行时权限，在该软件中没有实现。

2、播放模式。在开发中，一开始没有为播放器设置播放模式，当MediaPlayer 播放完当前音乐文件后，会自动停止。改进之后，当艺术歌曲播放完之后，会自动播放音乐列表中的下一首歌曲。不过，仍然没有增加可以选择播放模式的按钮。

3、歌曲扫描问题。该音乐播放器只是实现获取特定本地文件夹下的音乐文件，没有实现扫描本机内容或者 SD 卡文件的功能。

4、多个界面切换的问题。在播放页面返回主界面之后，在主界面没有查看当前播放音乐的接口，用户体验会差一些。事实上，在开发的过程中，我也尝试过当播放页面返回主界面时进行数据回调，在主界面底部显示当前播放音乐的信息，并提供跳转的接口，但是在实现过程中，主界面的 onActivityResult()一直无法接收到播放页面的回传数据，所以最后就没有实现该功能。

5、通知栏显示问题。在开发过程中，也设想过在标题栏添加一个通知，用于显示和控制当前正在播放的音乐，当时因为上一个问题没有解决，最后也没有实现通知栏。

6、手机来电问题。一般的音乐播放器在播放时，如果手机来电，会停止音乐播放，而该播放器还没有实现该功能。

六、代码设计：

程序主要代码：

```
/*************** 主界面：MainActivity.java ***************/
package com.example.mymusicplayer3;
```

```
import android.content.Intent;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;
import java.io.File;
import java.util.ArrayList;


public class MainActivity extends AppCompatActivity {
    private MyListView list_view;
    private ArrayList<String> fileNames;
    private int current_position;
    private String[] list;

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

        Initial();
    }

    private void Initial() {
        list_view = (MyListView) findViewById(R.id.list_view);

//        读取手机存储根目录下 Music 目录下的音乐文件
        getMusicList();
        updateListView();

        list_view.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
//                将音乐列表传给 Service
                list = new String[fileNames.size()];
                for (int i = 0; i < list.length; i++) {
                    list[i] = fileNames.get(i);
                }
```

```
                current_position = position;
                Intent intent = new Intent(MainActivity.this,
Player.class);
                intent.putExtra("list", list);
                intent.putExtra("position", current_position);
                startActivity(intent);
            }
        });
    }

    private void getMusicList() {
        fileNames = new ArrayList<String>();
        String path =
Environment.getExternalStorageDirectory().getAbsoluteFile() +"/Music";
        File directory = new File(path);
        if (directory.exists()) {
            for (File file : directory.listFiles()) {
                if (!file.isDirectory() &&
file.getName().endsWith(".mp3")) {
                    fileNames.add(file.getName());
                }
            }
        } else {
            Toast.makeText(this, "读取文件失败，请检查路径是否正确",
Toast.LENGTH_SHORT).show();
        }
    }

    private void updateListView() {
        MyBaseAdapter myAdapter = new MyBaseAdapter(this, fileNames);
        list_view.setAdapter(myAdapter);

//        ScrollView滚动至顶部
        ScrollView scrollView = (ScrollView)
findViewById(R.id.scrollView);
        scrollView.smoothScrollTo(0, 0);

        TextView tv_count = (TextView) findViewById(R.id.tv_count);
        String str_count = "（共" +  fileNames.size() +"首）";
        tv_count.setText(str_count);
    }
}
```

/**************** 播放界面：Player.java ****************/

```
package com.example.mymusicplayer3;

import android.app.Application;
import android.app.Service;
import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.graphics.Color;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import java.util.Timer;
import java.util.TimerTask;

public class Player extends AppCompatActivity implements
View.OnClickListener{
    private TextView tv_name;
    private SeekBar progress;
    private TextView tv_current_time;
    private TextView tv_total_time;
    private ImageView btn_pause_play;
    private ImageView btn_previous;
    private ImageView btn_next;

    private String[] list;
    private int position;

    private Timer timer;
    private Intent intent_music;
    private MyConn conn;
    MusicService.MyBinder binder;

    private Handler handler = new Handler();
```

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

        Initial();

        getData();

//        创建和绑定服务
        initialService();

        setListener();
    }

    private void Initial() {
        tv_name = (TextView) findViewById(R.id.tv_name);
        progress = (SeekBar) findViewById(R.id.progress);
        tv_current_time = (TextView)
findViewById(R.id.tv_current_time);
        tv_total_time = (TextView) findViewById(R.id.tv_total_time);
        btn_pause_play = (ImageView) findViewById(R.id.btn_pause_play);
        btn_previous = (ImageView) findViewById(R.id.btn_previous);
        btn_next = (ImageView) findViewById(R.id.btn_next);

        conn = new MyConn();
    }

    private void getData() {
        Intent intent = getIntent();
        list = intent.getStringArrayExtra("list");
        position = intent.getIntExtra("position", 0);
    }

    private void initialService() {
        intent_music = new Intent(this, MusicService.class);
        intent_music.putExtra("list", list);
        intent_music.putExtra("position", position);
        bindService(intent_music, conn, BIND_AUTO_CREATE);
    }

    private void setListener() {
        btn_pause_play.setOnClickListener(this);
```

```
        btn_previous.setOnClickListener(this);
        btn_next.setOnClickListener(this);

//        进度条滑动
        progress.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                binder.seekTo(progress.getProgress());
            }

            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                tv_current_time.setText(exchangeToTime(progress));
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }
        });
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_pause_play:
                if (binder.isPlaying()) {
                    binder.Pause();
                    cancelTimer();
                    btn_pause_play.setImageResource(R.mipmap.ic_play);
                }
                else {
                    binder.Play();
                    setTimer();

btn_pause_play.setImageResource(R.mipmap.ic_pause);
                }
                break;
            case R.id.btn_previous:
                binder.Previous();

tv_total_time.setText(exchangeToTime(binder.getMusicTotalTime()));
```

```
                tv_name.setText(binder.getMusicName());
                progress.setMax(binder.getMusicTotalTime());
                setTimer();
                break;
            case R.id.btn_next:
                binder.Next();

tv_total_time.setText(exchangeToTime(binder.getMusicTotalTime()));
                tv_name.setText(binder.getMusicName());
                progress.setMax(binder.getMusicTotalTime());
                btn_pause_play.setImageResource(R.mipmap.ic_pause);
                setTimer();
                break;
            default:
                break;
        }
    }

    private void setTimer() {
        cancelTimer();

        timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                handler.post(new Runnable() {
                    @Override
                    public void run() {

tv_current_time.setText(exchangeToTime(binder.getMusicCurrentTime()));

progress.setProgress(binder.getMusicCurrentTime());
                        if (binder.isStopped()) {
                            binder.Next();

tv_total_time.setText(exchangeToTime(binder.getMusicTotalTime()));
                            tv_name.setText(binder.getMusicName());

progress.setMax(binder.getMusicTotalTime());
                            setTimer();
                        }
                    }
                });
            }
```

```
        }, 0, 1000);
    }

    private void cancelTimer() {
        if (timer != null) {
            timer.cancel();
            timer = null;
        }
    }

    private class MyConn implements ServiceConnection {

        @Override
        public void onServiceConnected(ComponentName name, IBinder
service) {
            binder = (MusicService.MyBinder) service;
            binder.initialPlayer();

tv_total_time.setText(exchangeToTime(binder.getMusicTotalTime()));
            tv_name.setText(binder.getMusicName());
            progress.setMax(binder.getMusicTotalTime());
            setTimer();
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {

        }
    }

//    将时间长度以字符串输出
    private String exchangeToTime(int time) {
        String str_time = "";
        int minute, second;
        time = time / 1000;
        minute = time / 60;
        second = time % 60;
        if (minute < 10)
            str_time += "0";
        str_time += String.valueOf(minute) + ":";
        if (second < 10)
            str_time += "0";
        str_time += second;
        return str_time;
```

```
    }

    @Override
    protected void onDestroy() {
        unbindService(conn);
        super.onDestroy();
    }
}
```

/*************** 音乐服务：MusicService.java ***************/

```
package com.example.mymusicplayer3;

import android.app.Service;
import android.content.Intent;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Binder;
import android.os.Environment;
import android.os.IBinder;
import android.util.Log;

public class MusicService extends Service {
    private String[] list;
    private int position;
    private MediaPlayer media_player;

    private boolean is_playing = false;
    private boolean is_stopped = false;
    private int current_time;
    private int total_time;
    private String name;

    public MusicService() {
    }

    class MyBinder extends Binder {
        public void initialPlayer() {
            try {
//              停止当前正在播放的音乐
                if (media_player != null) {
                    media_player.stop();
```

```
                    media_player.release();
                    media_player = null;
                }

                String path =
Environment.getExternalStorageDirectory().getAbsolutePath() +
"/Music/";
                path += list[position];
//                media_player = new MediaPlayer();
//
media_player.setAudioStreamType(AudioManager.STREAM_MUSIC);
//                media_player.setDataSource(path);
//                media_player.prepare();
                media_player = MediaPlayer.create(MusicService.this,
Uri.parse(path));
                media_player.start();
                is_stopped = false;
                is_playing = true;
                total_time = media_player.getDuration();
                name = list[position].substring(0,
list[position].lastIndexOf("."));
                media_player.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
                    @Override
                    public void onCompletion(MediaPlayer mp) {
                        is_stopped = true;
                    }
                });
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        public void Pause() {
            if (media_player != null && media_player.isPlaying()) {
                media_player.pause();
            }
            is_playing = false;
        }

        public  void Play() {
            if (media_player != null && !media_player.isPlaying()) {
                media_player.start();
            }
```

```
            is_playing = true;
        }

        public boolean isPlaying() {
            return is_playing;
        }

        public boolean isStopped() {
            return is_stopped;
        }

        public void Previous() {
            position = (position + list.length - 1) % list.length;
            initialPlayer();
        }

        public void Next() {
            position = (position + 1) % list.length;
            initialPlayer();
        }

        public int getMusicCurrentTime() {
            if (media_player != null) {
                current_time = media_player.getCurrentPosition();
            }
            return current_time;
        }

        public int getMusicTotalTime() {
            return total_time;
        }

        public String getMusicName() {
            return name;
        }

        public void seekTo(int position) {
            media_player.seekTo(position);
        }

    }

    @Override
    public void onCreate() {
```

```
        Log.i("service", "onCreate");
        super.onCreate();
    }

    @Override
    public IBinder onBind(Intent intent) {
        list = intent.getStringArrayExtra("list");
        position = intent.getIntExtra("position", 0);
        return new MyBinder();
    }

    @Override
    public boolean onUnbind(Intent intent) {
        return super.onUnbind(intent);
    }

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