



大连理工大学  
DALIAN UNIVERSITY OF TECHNOLOGY

# 《KK 音乐——小型音乐播放 App》

## 设计报告

班 级: 管物 1901

学 号: 201903020

学 生 姓 名: 严梓锴

指 导 教 师: 卢涛 王旭坪

完 成 日 期: 2021.12.23

大连理工大学

Dalian University of Technology

## 目 录

1	App 介绍 .....	1
1.1	基本介绍 .....	1
1.2	技术要求 .....	1
1.2.1	基本技术要求 .....	1
1.2.2	额外技术要点 .....	1
1.3	界面展示 .....	2
2	功能分析 .....	2
2.1	功能一：注册 .....	2
2.2	功能二：登录及自动登录 .....	2
2.3	功能三：基本信息查看功能 .....	3
2.4	功能四：退出登录及通知 .....	3
2.5	功能五：注销登录状态 .....	3
2.6	功能六：歌手查询功能 .....	3
2.7	功能七：打开音乐播放器 .....	3
2.8	功能八：音乐播放、暂停、继续播放、退出 .....	4
2.9	功能九：音乐播放进度拖拽 .....	4
3	程序结构 .....	5
3.1	程序总体结构 .....	5
3.2	Activity 程序结构 .....	5
3.2.1	LoginActivity(登录界面) .....	5
3.2.2	OperateActivity(主界面) .....	6
3.2.3	MusicActivity(音乐播放界面) .....	6
3.3	Service 程序结构 .....	7
3.3.1	MusicPlayerService(主界面) .....	7
4	系统实现 .....	8
4.1	界面的实现 .....	8
4.1.1	界面一：登陆界面 .....	8
4.1.2	界面二：App 主界面 .....	8
4.1.3	界面三：音乐播放界面 .....	9
4.2	关键技术的实现 .....	9

4.2.1	注册的实现——应用数据存储 .....	9
4.2.2	登录的实现——应用界面跳转、参数传递、数据存储 .....	10
4.2.3	回显数据的实现——应用数据存储 .....	11
4.2.4	自动登录的实现——应用数据存储、参数传递 .....	11
4.2.5	用户基本信息展示的实现——应用参数传递 .....	12
4.2.6	歌手查询的实现 .....	12
4.2.7	跳转音乐播放界面的实现——应用界面转跳和参数传递 .....	13
4.2.8	注销的的登录状态的实现——应用 Broadcast .....	14
4.2.9	退出登录并发送通知的实现——应用 Notification .....	15
4.2.10	播放音乐、暂停播放、继续播放、退出播放的实现——应用 Service .....	16
4.2.11	音乐播放进度拖拽的实现 .....	19
4.2.12	音乐播放时打碟（碟片转动）的实现 .....	20
总 结 .....	.....	21

## 1 App 介绍

### 1.1 基本介绍

KK 音乐为一个小型音乐播放器。该 App 一共有三个界面，分别为登录界面、App 主界面以及音乐播放界面。

该 App 实现的主要业务功能有用户的登录与退出、用户数据的储存、歌手查询以及音乐播放等。

### 1.2 技术要求

#### 1.2.1 基本技术要求

该 App 实现了以下基本技术要求。

- (1) 具有两个以上的界面，能够实现界面跳转和参数传递。
- (2) 包含按钮、显示文本框、输入文本框、下拉列表、单选框等四种以上的控件。
- (3) 应用了 Notification、Broadcast 和 Service。
- (4) 应用了数据存储。

#### 1.2.2 额外技术要点

该 App 额外实现了以下技术要点。

- (1) 使用了控件 MediaPlayer 实现音乐的播放。
- (2) 使用了控件 ImageView 插入了唱片图片。
- (3) 使用了控件 SeekBar 实现音乐进度条的显示及拖拽进度。
- (4) 使用了 Timer 类对所播放的音乐进行记时。
- (5) 使用了 ObjectAnimator 类实现了唱片的旋转与暂停。
- (6) 更换了 App 的图标设计以及部分页面的背景和控件的设计。

### 1.3 界面展示

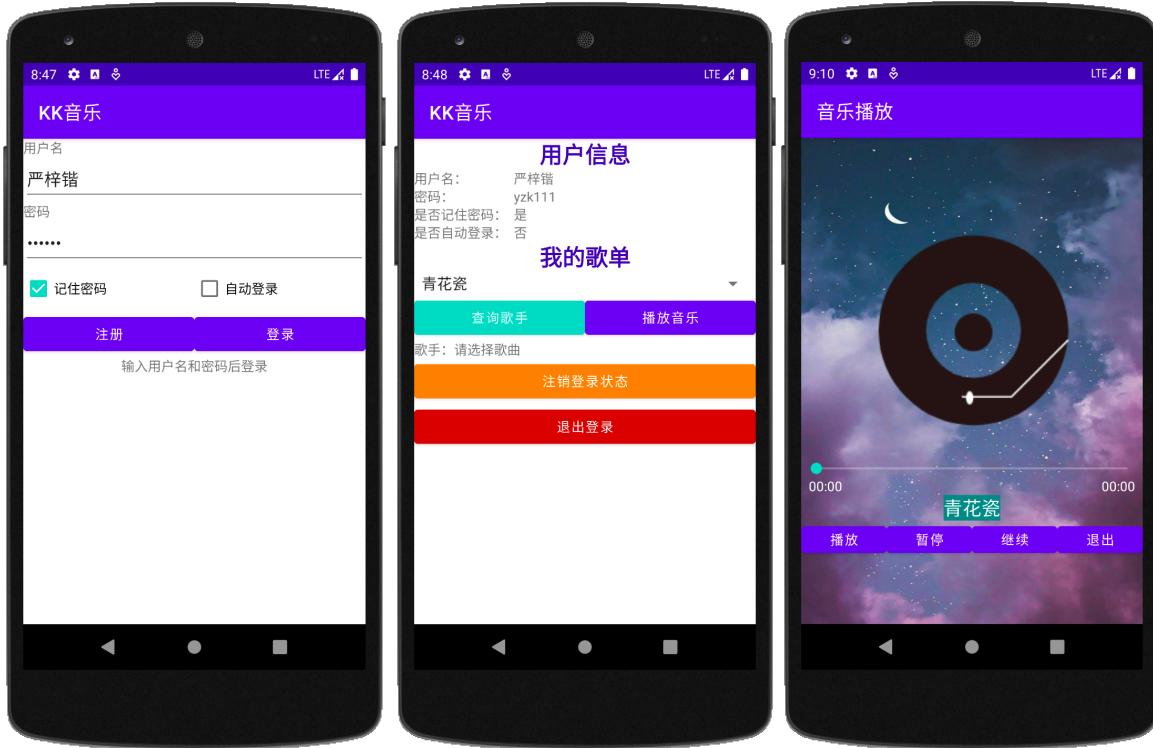


图 1 App 界面图（从左至右依次为：登录界面、App 主界面、音乐播放界面）

## 2 功能分析

### 2.1 功能一：注册

首次打开 App 进入登陆界面，用户在登录界面输入用户名和密码，输入完成后可以选择记住密码和自动登录的选项，点击“注册”按钮即可注册。若注册成功则回在界面上文字提示用户“注册成功”。

当用户注册时输入的用户名或密码为空时拒绝用户的注册操作，并在界面上用文字提示用户“用户名或密码为空”，无法进行注册。

用户注册完成后，用户名、密码、是否选择记住密码、是否选择自动登录四个基本用户数据将会被存储下来。

### 2.2 功能二：登录及自动登录

在 App 的登录界面中，当 App 已有用户的注册数据后，用户可以进行登录操作。输入用户名和密码，点击“登录”按钮进行登录。App 会检查用户名和密码是否与已注

册的数据匹配。如果匹配，则允许用户登录，在界面上文字提示用户“登录成功”，并转跳页面至 App 主界面；若不匹配，则文字提示用户“用户名或密码不正确”，拒绝用户的登录行为。

若用户在某次登录时选择“自动登录”，则当用户下一次重新打开该 App 后，App 将会执行自动登录操作。

### 2.3 功能三：基本信息查看功能

在 App 主界面顶端有对于当前用户基本信息的显示。包括用户名、密码、是否记住密码、是否自动登录。用户可以在本界面中及时看到界面基本信息。

### 2.4 功能四：退出登录及通知

在 App 主界面中，已登录的用户可以点击“退出登录”按钮进行退出登录。退出登录后，App 界面会返回到登陆界面。

用户点击“退出登录”的同时，App 会发送通知，通知名为“您已退出 KK 音乐”，通知内容为“点击此返回登陆界面”。无论手机处在哪个页面，点击该通知后都能再次跳转到 App 的登录界面。

### 2.5 功能五：注销登录状态

在 App 主界面中，已登录的用户可以点击“注销登录状态”按钮进行登录状态的注销。登录状态注销后，页面顶端显示的用户基本信息将全部变为默认的空值，即当前为无用户登录的状态。

此时，App 的页面仍保留在 App 的主界面上，用户仍然可以执行主界面上的任何操作，只是在“用户信息”栏中没有对与用户信息的显示。

### 2.6 功能六：歌手查询功能

在 App 主界面中，用户可以在“我的歌单”下方的下拉列表中选择不同的歌曲名字，选定某一首歌曲后，点击下方的按钮“查询歌手”，在下方的文本框中会呈现出该首歌的原唱歌手名。

### 2.7 功能七：打开音乐播放器

在 App 的主界面中，用户首先在“我的歌单”下方的下拉列表中选择想要播放的音乐，选择完成后点击下方“播放音乐”按钮，此时 App 的界面会跳转到音乐播放界面。在音乐播放界面上会显示当前歌曲的歌曲名。

## 2.8 功能八：音乐播放、暂停、继续播放、退出

在音乐播放界面中，用户点击“播放”按钮可以开始音乐的播放。同时，界面中的“唱片”图片将会开始旋转。

当用户点击“暂停”按钮后，正在播放的音乐将会暂停播放，并且“唱片”图片将会带当前位置停止旋转。

当用户点击“继续”按钮后，音乐会继续进行播放，同时“唱片”图片将会继续从当前位置开始旋转。

当用户点击“退出”按钮后，音乐播放界面将会被关闭。

在音乐播放或暂停的任何时刻，当用户在此点击“播放”按钮后，音乐将会从头开始播放。并且“唱片”将会回到最初的位置，重新开始旋转。

## 2.9 功能九：音乐播放进度拖拽

在音乐播放界面中，当音乐开始播放时，“唱片”下方的“进度条”将会同步开始跟进进度。进度条的左下角显示当前播放的进度，进度条的右下角显示所播放的音乐的总时长。

用户可以直接拖拽进度条到想要到的位置，音乐播放的进度也会相应进行更新。

### 3 程序结构

#### 3.1 程序总体结构

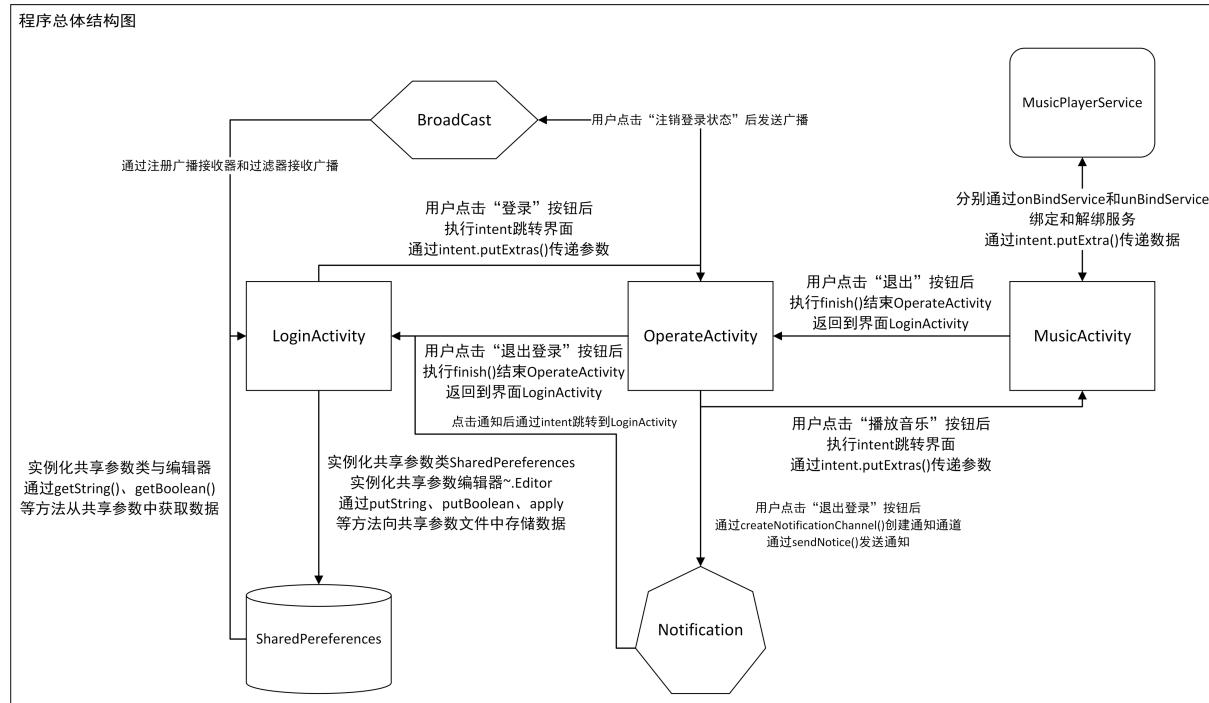


图 2 程序总体结构图

#### 3.2 Activity 程序结构

##### 3.2.1 LoginActivity(登录界面)

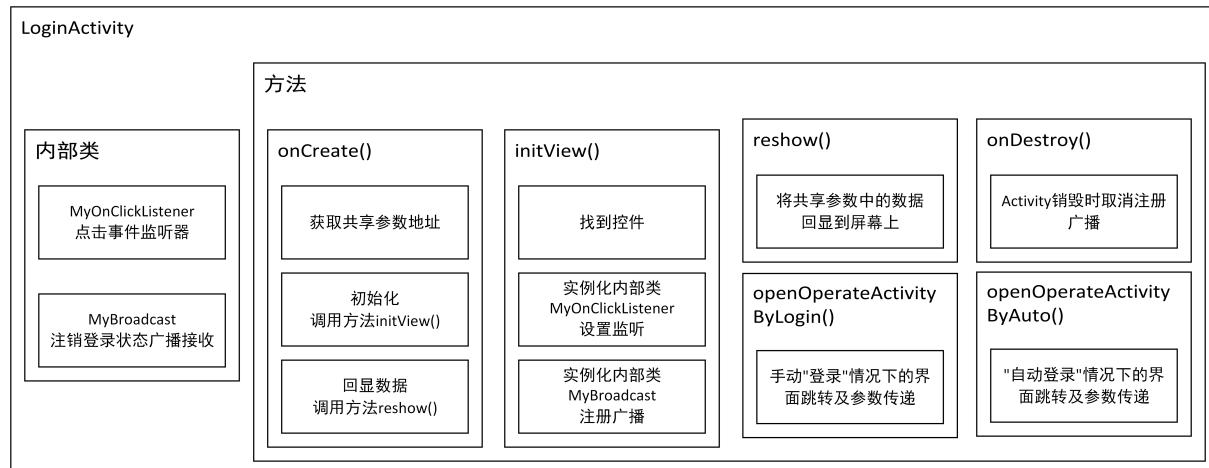


图 3 LoginActivity(登录界面)程序结构

### 3.2.2 OperateActivity(主界面)

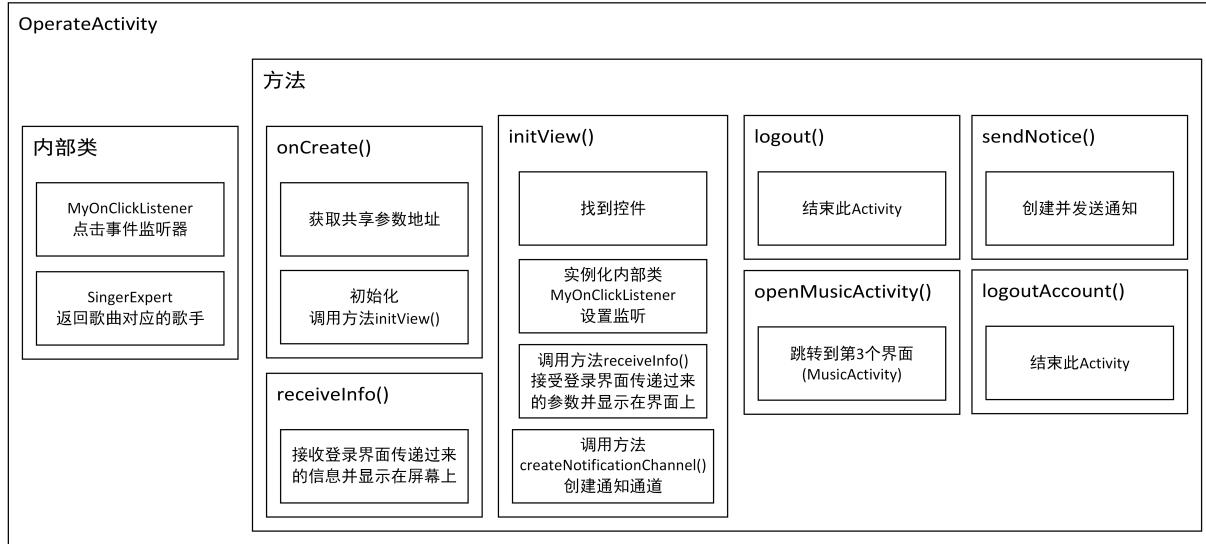


图 4 OperateActivity(主界面)程序结构

### 3.2.3 MusicActivity(音乐播放界面)

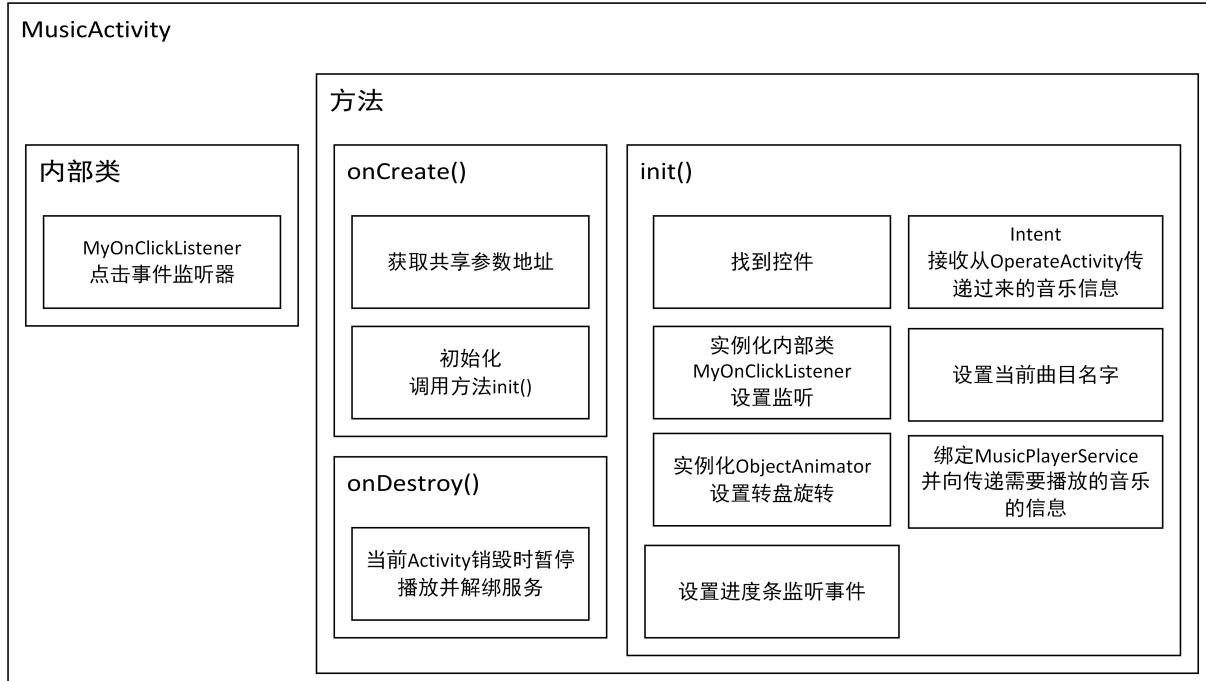


图 5 MusicActivity(音乐播放界面)程序结构

### 3.3 Service 程序结构

#### 3.3.1 MusicPlayerService(主界面)

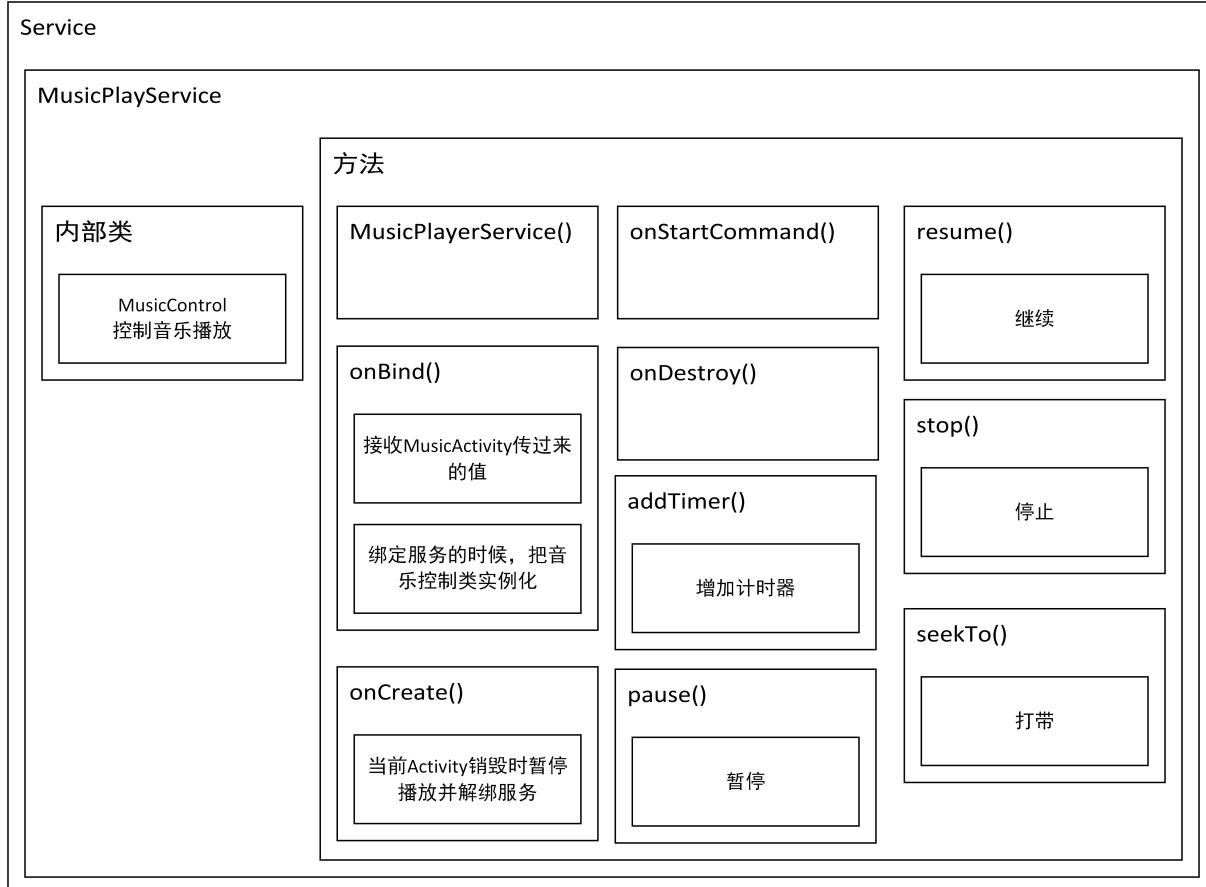


图 6 MusicPlayerService(主界面)程序结构

## 4 系统实现

### 4.1 界面的实现

#### 4.1.1 界面一：登陆界面

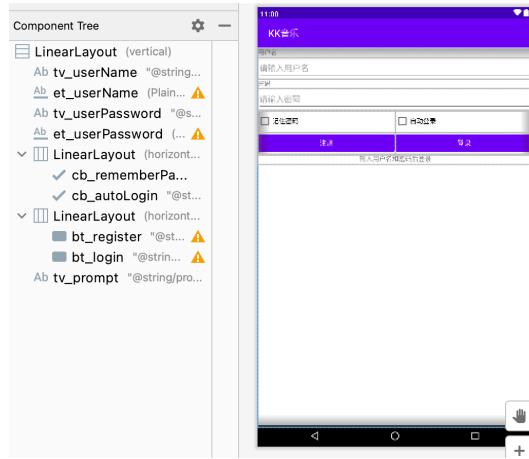


图 7 登录界面页面布局

#### 4.1.2 界面二：App 主界面

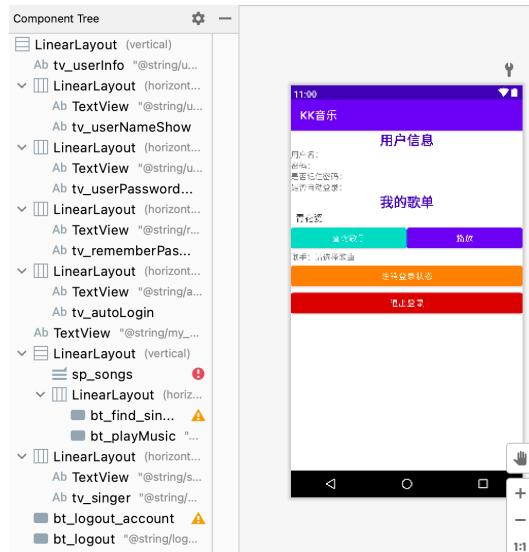


图 8 App 主界面页面布局

#### 4.1.3 界面三：音乐播放界面

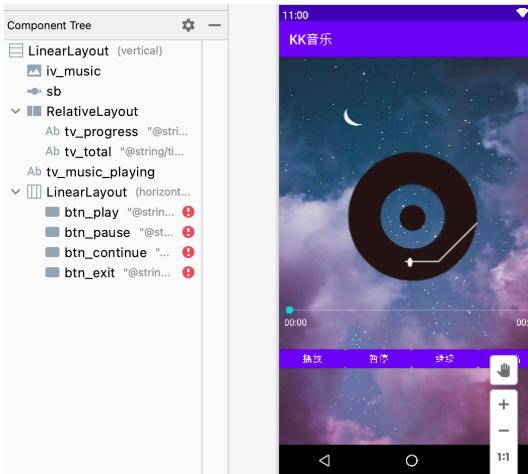


图 9 音乐播放界面页面布局

### 4.2 关键技术的实现

#### 4.2.1 注册的实现——应用数据存储

(1) 在 LoginActivity 中创建点击事件监听器类 MyOnClickListener

```
private class MyOnClickListener implements View.OnClickListener {...}
```

(2) 在监听器类中重写点击事件 onClick(), 在其中设置对“注册”按钮点击的响应操作

判断点击的是否为“注册按钮”，如果是，则对输入文本框“用户名”和“密码”中的输入项进行提取并转换为字符串，同时检查“记住密码”和“自动登录”选项框是否被勾选。

进一步判断所输入的“用户名”和“密码”是否为空值，若都不为空，则获取共享参数，将上述四项数据添加到共享参数（SharedPreference）中。否则，拒绝用户的注册请求，并给出文字提示“用户名或密码为空”。

以下为用户名和密码都满足要求并勾选了“记住密码”和“自动登录”选项后的关键代码。

```
SharedPreferences.Editor editor = sp.edit();
editor.putString("name", name);
editor.putString("password", password);
editor.putBoolean("rememberPassword", true);
editor.putBoolean("autoLogin", true);
editor.apply();
```

(3) 实例化监听器，对“注册”按钮设置监听事件

在初始化过程中，实例化监听器，并对“注册”按钮设置监听事件。

---

```
MyOnClickListener l = new MyOnClickListener();  
bt_register.setOnClickListener(l);
```

---

#### 4.2.2 登录的实现——应用界面跳转、参数传递、数据存储

##### (1) 在 LoginActivity 中的监听器类中设置对“注册”按钮点击的响应操作

当用户点击“登录”按钮后，首先获取“用户名”和“密码”两个输入文本框中的输入内容，并转换为字符串。

其次，获取共享参数文件中所储存的用户名、密码，将其与用户输入的值进行比较，如果两者均相等匹配，提示用户登录成功，并调用方法 openOperateActivityByLogin()，转跳至第二个界面（OperateActivity）。

如果用户登录时勾选了记住密码或自动登录，则将其记录在共享参数（SharedPreference）中。

关键部分代码如下所示（“...”表示代码的省略）。

---

```
String loginName = et_userName.getText().toString().trim();  
String loginPassword = et_userPassword.getText().toString().trim();  
if (loginName.equals(sp.getString("name", null)) &&  
    loginPassword.equals(sp.getString("password", null))) {...} else {...}
```

---

##### (2) 界面跳转及参数传递方法 openOperateActivityByLogin() 的实现

该方法包含从当前登陆界面跳转到 App 主界面的意图 intent，以及传递相关用户信息参数。代码如下所示。

---

```
private void openOperateActivityByLogin() {  
    // 从 sp 中调取用户信息数据，并打包放入 Bundle 数据类型中  
    Bundle bd = new Bundle();  
    bd.putString("name", et_userName.getText().toString().trim());  
    bd.putString("password", et_userPassword.getText().toString().trim());  
    bd.putBoolean("rememberPassword", cb_rememberPassword.isChecked());  
    bd.putBoolean("autoLogin", cb_autologin.isChecked());  
    // 定义一个意图，将 Bundle 类型中的数据传给这个意图，并实现页面的跳转  
    Intent intent = new Intent();  
    intent.putExtras(bd);  
    intent.setClass(LoginActivity.this, OperateActivity.class);  
    startActivity(intent);  
}
```

---

##### (3) 实例化监听器，对“登录”按钮设置监听事件

对“注册”按钮设置监听事件。

---

```
MyOnClickListener l = new MyOnClickListener();  
bt_login.setOnClickListener(l);
```

---

#### 4.2.3 回显数据的实现——应用数据存储

##### (1) 在 LoginActivity 中定义回显数据方法 reshaw()

```
private void reshaw() {
    boolean rememberPassword = sp.getBoolean("rememberPassword", false);
    boolean autologin = sp.getBoolean("autoLogin", false);
    // 如果之前勾选了"记住密码"
    if (rememberPassword) {
        // 获取 SP 里面的 name 和 password, 并保存到 EditText 里面
        String name = sp.getString("name", "");
        String password = sp.getString("password", "");
        et_userName.setText(name);
        et_userPassword.setText(password);
        cb_rememberPassword.setChecked(true); // "记住密码"打上勾
    }
    // 如果之前勾选了"自动登录"
    if (autologin) {
        cb_autologin.setChecked(true); // "自动登录"打上勾
        // 模拟自动登录
        Toast.makeText(this, "自动登录了", Toast.LENGTH_SHORT).show();
        // 自动转跳到第二个界面(OperateActivity)
        openOperateActivityByAuto();
    }
}
```

---

##### (2) 在 onCreate 方法中调用回显数据的方法

```
// 回显数据: 再次打开时候, 从 SP 获取数据, 进行画面的同步
reshaw();
```

---

#### 4.2.4 自动登录的实现——应用数据存储、参数传递

##### (1) 在 LoginActivity 中定义自动登录时专挑界面的方法

```
// "自动登录"情况下的界面跳转及参数传递
private void openOperateActivityByAuto() {
    // 从 sp 中调取用户信息数据, 并打包放入 Bundle 数据类型中
    Bundle bd = new Bundle();
    bd.putString("name", sp.getString("name", ""));
    bd.putString("password", sp.getString("password", ""));
    bd.putBoolean("rememberPassword", sp.getBoolean("rememberPassword",
false));
    bd.putBoolean("autoLogin", sp.getBoolean("autoLogin", false));
    // 定义一个意图, 将 Bundle 类型中的数据传给这个意图, 并实现页面的跳转
    Intent intent = new Intent();
    intent.putExtras(bd);
    intent.setClass(LoginActivity.this, OperateActivity.class);
```

```
        startActivity(intent);  
    }
```

---

(2) 在回显数据方法中调用 openOperateActivityByAuto()

在回显数据时判断是否之前勾选了“自动登录”，如果是则在回显数据方法中调用。

该部分代码已经在 4.2.3 回显数据部分进行描述和呈现。

#### 4.2.5 用户基本信息展示的实现——应用参数传递

(1) 在 OperateActivity 中创建接收 LoginActivity 传递过来的信息的方法

通过实例化 Intent 并接收 getIntent() 来接收 LoginActivity 传递过来的参数。

通过实例化 Bundle 类和 intent.getExtras() 取出传递过来的参数。

经过条件判断后，用对各文本框控件应用 setText() 方法来显示用户信息。

核心代码如下所示。

```
private void receiveInfo() {  
    Intent intent = getIntent();  
    Bundle bd = intent.getExtras();  
    // 之后通过 if-else 判断和 setText() 方法进行呈现，代码略
```

---

(2) 在 OperateActivity 的 onCreate 方法中调用 receiveInfo()

```
// 接受登录界面传递过来的参数并显示在界面上  
receiveInfo();
```

---

#### 4.2.6 歌手查询的实现

(1) 在 OperateActivity 中创建静态内部类 SingerExpert，用于判断并返回某一首歌的歌手。

```
private static class SingerExpert {  
    public String getSinger(String song) {  
        String result;  
        switch (song) {  
            case "青花瓷":  
                result = "周杰伦";  
                break;  
            case "江南":  
                result = "林俊杰";  
                break;  
            case "泡沫":  
                result = "邓紫棋";  
                break;  
            default:  
                result = "查不到这首歌的歌手";  
        }  
    }  
}
```

```
        return result;
    }
}
```

(2) 找到按钮控件“查询歌手”、用于显示歌手的文本框控件以及显示歌曲列表的下拉列表框。

(3) 创建监听器类 MyOnClickListener 监听点击事件，该类继承自 View.OnClickListener。

(4) 在监听器类 MyOnClickListener 中重写 onClick()方法，并在其中定义点击“查询歌手”按钮后的操作。

首先获取我的歌单下方的歌曲下拉列表中选中的歌曲，其次通过 SingerExpert 的方法 getSinger() 判断该首歌的歌手，并将歌手名显示在用于显示歌手的文本框控件中。

(5) 在初始化方法 initView 中实例化 MyOnClickListener，并在“查询歌手”按钮上设置点击监听。在 OperateActivity 的 onCreate() 方法中调用 initView()。

以上部分关键代码与前文所述的几个技术实现中的代码结构类似，故省略这部分非核心代码块。

#### 4.2.7 跳转音乐播放界面的实现——应用界面转跳和参数传递

(1) 在 OperateActivity 中找到按钮控件“播放音乐”以及显示歌曲的下拉列表框。

(2) 创建 openMusicActivity() 方法。

通过实例化 intent 和方法 setClass() 实现页面跳转的定义。

获取我的歌单下方的歌曲下拉列表中选中的歌曲，并将歌曲名字符串通过 intent.putExtra() 添加到 intent 附带的参数中。

代码如下所示。

```
private void openMusicActivity() {
    Intent intent = new Intent();
    String selectedMusic = sp_songs.getSelectedItem().toString();
    intent.putExtra("selected_music", selectedMusic);
    intent.setClass(this, MusicActivity.class);
    startActivity(intent);
}
```

(3) 在监听器类 MyOnClickListener 重写的 onClick() 方法中，添加点击“播放音乐”按钮后的操作，即调用 openMusicActivity() 方法。

(4) 在初始化方法 initView 中实例化 MyOnClickListener，并在“播放音乐”按钮上设置点击监听。在 OperateActivity 的 onCreate() 方法中调用 initView()。

#### 4.2.8 注销的的登录状态的实现——应用 Broadcast

(1) 在 OperateActivity 中找到按钮控件“注销登录状态”。

(2) 定义广播 ID 字符串常量

---

```
private final String BROADCAST_ID="dut.sem.zettayan.bigwork.broadcast";
```

---

(3) 创建 logoutAccount()方法。

首先实例化 intent 并传入广播 ID。其次通过本地广播管理器 LocalBroadcastManager 发送广播。文字提示用户登录状态已经被注销。在共享参数中将“自动登录”状态对应的值设为“false”。将用户信息设为空。

核心代码如下所示。

---

```
private void logoutAccount() {  
    // 发送注销登录状态的广播  
    Intent intent = new Intent(BROADCAST_ID);  
    LocalBroadcastManager.getInstance(this).sendBroadcast(intent);  
    // 提示已经注销登录状态  
    Toast.makeText(this, "已注销", Toast.LENGTH_SHORT).show();  
    // 如果之前勾选了"自动登录", 则取消自动登录的设置,防止返回后有跳转回此页面  
    boolean autologin = sp.getBoolean("autoLogin", false);  
    if (autologin){  
        SharedPreferences.Editor editor = sp.edit();  
        editor.putBoolean("autoLogin", false);  
        editor.apply();  
    }  
    // 将页面中的用户信息置空  
    userNameShow.setText("");  
    userPasswordShow.setText("");  
    rememberPassword.setText("");  
    autoLogin.setText("");  
}
```

---

(4) 在监听器类 MyOnClickListener 重写的 onClick()方法中，添加点击“注销登录状态”按钮后的操作，即调用 logoutAccount()方法。

(5) 在 LoginActivity 中创建 MyBroadCast 类，该类继承自广播接收器 BroadCastReceiver。

在该类中，定义操作。若果接收到的广播 ID 为 OperateActivity “注销登录状态”事件发生时的广播的 ID，则修改登录界面的提示信息为“已注销登录，请重新登录”。

---

```
private class MyBroadcast extends BroadcastReceiver{  
    @Override  
    public void onReceive(Context context, Intent intent) {
```

```
switch (intent.getAction()) {
    case ("dut.sem.zettayan.bigwork.broadcast"):
        tv_prompt.setText("已注销登录, 请重新登录");
        break;
}
}
```

---

- (6) 在初始化方法中实例化（注册）广播，通过 IntentFilter 过滤广播，通过 LocalBroadcastManager 接收该广播。

```
myBroadcast = new MyBroadcast();
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction("dut.sem.zettayan.bigwork.broadcast");
LocalBroadcastManager.getInstance(this).registerReceiver(myBroadcast,
    intentFilter);
```

---

- (7) 重写 LoginActivity 的 onDestroy 方法。当 Activity 销毁时取消注册广播

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

    LocalBroadcastManager.getInstance(this).unregisterReceiver(myBroadcast);
}
```

---

#### 4.2.9 退出登录并发送通知的实现——应用 Notification

- (1) 在 OperateActivity 中找到按钮控件“退出登录”。

- (2) 定义通知通道字符串常量以及通知 id。

代码如下。

```
private final String CHANNEL_ID="ChannelID";
private final int notificationId=1;
```

---

- (3) 创建 logout()方法。

代码如下。

在共享参数中将“自动登录”设置为“false”。

调用结束此 Activity 的方法 finish()。

```
private void logout() {
    // 如果之前勾选了“自动登录”，则取消自动登录的设置，防止返回后又跳转回此页面
    boolean autologin = sp.getBoolean("autoLogin", false);
    if (autologin){
        SharedPreferences.Editor editor = sp.edit();
        editor.putBoolean("autoLogin", false);
        editor.apply();
    }
}
```

---

```
// 结束此 Activity  
finish();  
}
```

(4) 创建新建通知通道的方法 `createNotificationChannel()` (代码同安卓开发者手册中给出的标准代码，此处略去呈现)。

(5) 在初始化方法中调用方法 `createNotificationChannel()` 创建通知通道。

(6) 创建发送通知的方法 `sendNotice()`。

用 `setContentTitle()` 设置通知的标题为“您已退出 KK 音乐”。

用 `setContentText()` 设置通知的内容为“点击此返回登录页面”。

实例化 Intent，响应该 Notification 的点击操作，传入参数 `LoginActivity.class`，使得点击该通知后跳转到登录页面。通过实例化 PendingIntent 实现该跳转。

代码如下所示。

```
private void sendNotice() {  
    Intent intent = new Intent(this, LoginActivity.class);  
    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent,  
0);  
    NotificationCompat.Builder builder = new  
NotificationCompat.Builder(this, CHANNEL_ID)  
        .setSmallIcon(R.mipmap.ic_launcher)  
        .setContentTitle("您已退出 KK 音乐")  
        .setContentText("点击此返回登录页面")  
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)  
        .setContentIntent(pendingIntent)  
        .setAutoCancel(true);  
    NotificationManagerCompat notificationManager =  
NotificationManagerCompat.from(this);  
    notificationManager.notify(notificationId, builder.build());  
}
```

(7) 找到按钮“退出登录”，在监听器类 `MyOnClickListener` 的 `onClick` 方法中添加对于按钮“退出登录”的操作，即调用 `logout()` 方法和 `sendNotice()` 方法。

#### 4.2.10 播放音乐、暂停播放、继续播放、退出播放的实现——应用 Service

(1) 创建 Service——`MusicPlayService`

① 声明多媒体对象 `MediaPlayer`，声明字符串对象用于接收传递过来的歌曲名

```
private MediaPlayer mediaPlayer; // 设置多媒体对象  
private String selectedMusic; // 选定的需要播放的音乐
```

② 定义 `MusicControl` 类，该类继承自 `Binder` 父类，用于控制音乐的播放。

在类中定义方法 play(), 用于启动音乐的播放。通过 switch 语句判断从 OperateActivity 接受来的歌曲名，并做对应的后续播放操作。

三条关键的实现播放功能的代码如下所示。

```
mediaPlayer.reset();
mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.qinghuaci);
mediaPlayer.start();
```

该类中定义方法 pause(), 暂停当前歌曲的播放。

在方法体中调用如下语句。

```
mediaPlayer.pause()
```

在类中定义方法 resume(), 继续当前音乐的播放。

在方法体中调用如下语句。

```
mediaPlayer.start(); // 播放, 不会重置
```

在类体中定义方法 stop(), 停止当前音乐的播放。

在方法体中调用如下语句。

```
public void stop() {
    mediaPlayer.stop(); //停止音乐
    mediaPlayer.release();
    try {
        timer.cancel();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

在类体中定义方法 seekTo(), 实现打带功能。

在方法体中调用如下语句。

```
public void seekTo(int ms) {
    mediaPlayer.seekTo(ms);
}
```

### ③ 重写 onBind 方法。

用 getStringExtra()方法获得 MusicActivity 传递过来的歌名。

返回 MusicControl，用于绑定服务的时候，把音乐控制类实例化。

```
@Override
public IBinder onBind(Intent intent) {
    selectedMusic = intent.getStringExtra("selected_music");
    return new MusicControl(); // 绑定服务的时候, 把音乐控制类实例化
}
```

### ④ 重写 onCreate 方法，实例化多媒体对象

```
@Override
public void onCreate() {
```

```
super.onCreate();
MediaPlayer = new MediaPlayer();
}
```

---

- (2) 在 MusicActivity 中声明并获取所有的控件  
(3) 声明一个 MusicPlayerService 中的 MusicControl 对象为 control

```
private MusicPlayerService.MusicControl control;
```

---

- (4) 实例化一个 ServiceConnection 对象 conn。  
① 重写方法 onServiceConnected(), 实例化对象 control

```
control = (MusicPlayerService.MusicControl) service;
```

---

- (5) 用 getIntent 接收从 OperateActivity 传递过来的音乐信息

```
Intent getIntent = getIntent();
String selectedMusic = getIntent.getStringExtra("selected_music");
```

---

- (6) 设置当前曲目的名字

```
tv_music_playing.setText(selectedMusic);
```

---

- (7) 通过 Intent 和 bindService 绑定 MusicplayerService，并传递需要播放的信息

```
Intent intent = new Intent(getApplicationContext(),
MusicPlayerService.class);
intent.putExtra("selected_music", selectedMusic);
bindService(intent, conn, BIND_AUTO_CREATE);
```

---

- (8) 设置监听器类 MyOnClickListener，该类继承自 View.OnClickListener。  
重写方法 onClick(), 通过 switch 语句判断哪个按钮控件触发了点击事件。  
对“播放”“暂停”“继续”“推出”四个按钮分别调用对象 control 中对应语义的方法。  
(9) 在 onCreate 中实例化 MyOnClickListener，并在“播放”“暂停”“继续”“推出”  
四个按钮控件上分别设置监听。  
(10) 重写 onDestroy 方法，当前 Activity 销毁时调用方法 stop() 和 unbindService(), 暂停  
播放并解绑服务。

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

---

#### 4. 2. 11 音乐播放进度拖拽的实现

(1) 在 MusicPlayerService 中创建增加计时器方法 addTimer()。

实例化一个 Timer 类的对象 timer。

实例化一个 TimerTask 对象 task，在其中重写方法 run()，在该方法中利用 mediaPlayer 的各类方法完成以下操作：获取歌曲总时长、获取当前播放进度、创建消息对象、将音乐的总时长和播放进度封装至消息对象中、将消息发送到主线程的消息队列。

利用方法 schedule() 设定开始计时任务后的 5 毫秒，第一次执行 task 任务，以后每 500 毫秒执行一次。

(2) 在 MusicPlayerActivity 的 onCreate 方法中设置进度条监听事件。

---

```
musicProgressBar.setOnSeekBarChangeListener(new  
SeekBar.OnSeekBarChangeListener())
```

---

重写其中的 onProgressChanged() 方法。

---

```
//当音乐停止后，停止光盘动画  
if (progress == seekBar.getMax()) {  
    animator.pause();  
}  
if (fromUser) { // 判断是不是用户拖动的  
    control.seekTo(progress);
```

---

重写其中的 onStartTrackingTouch() 方法。

---

```
control.pause();
```

---

重写其中的 onStopTrackingTouch() 方法。

---

```
control.resume();
```

---

(3) 实例化一个 Handler 类为 handler，实现获取从 MusicPlayerService 传递过来的消息的功能。

---

```
public static Handler handler = new Handler(Looper.myLooper())
```

---

重写其中的 handleMessage 方法。

---

```
super.handleMessage(msg);  
Bundle bundle = msg.getData(); // 获取从子线程发送过来的音乐播放进度  
int duration = bundle.getInt("duration"); // 歌曲的总时长  
int currentPosition = bundle.getInt("currentPosition"); // 歌曲当前进度  
musicProgressBar.setMax(duration); // 设置 Seekbar 的最大值为歌曲总时长  
musicProgressBar.setProgress(currentPosition); // 设置 Seekbar 当前的进度位置  
String totalTime = msToMinSec(duration); // 歌曲总时长  
String currentTime = msToMinSec(currentPosition); // 歌曲当前播放时长  
totalTv.setText(totalTime);  
currentTv.setText(currentTime);
```

---

(4) 定义一个时间转换方法，使得音乐播放的实践能够以“分:秒”的形式呈现。

```
public static String msToMinSec(int ms) {  
    int sec = ms / 1000;  
    int min = sec / 60;  
    sec -= min * 60;  
    return String.format("%02d:%02d", min, sec);  
}
```

---

#### 4.2.12 音乐播放时打碟（碟片转动）的实现

(1) 在 MusicActivity 中实例化一个对象 ObjectAnimator 为 animator。

(2) 在 onCreate() 方法中设置转盘旋转。

```
// 设置转盘旋转  
animator = ObjectAnimator.ofFloat(iv_disk, "rotation", 0, 360.F); // 对象是  
iv_disk, 动作是 rotation 旋转, 角度从 0 到 360 度, 这里用的是浮点数, 所以要加个 F  
animator.setDuration(10000); // 这是设置动画的时长, 单位为毫秒, 这里设置了 10 秒转一圈  
animator.setInterpolator(new LinearInterpolator()); // 旋转时间函数为线性, 意为  
匀速旋转  
animator.setRepeatCount(-1); // 设置转动圈数, -1 为一直转动
```

---

(3) 设置进度条监听事件 (4.2.11 中已经呈现)

(4) 在点击监听器类 MyOnClickListener 中对于每个按钮的点击事件，根据语义，对 animator 执行相应的操作。分别包含以下代码。

```
// 光盘开始转动  
animator.start();  
  
// 光盘暂停  
animator.pause();  
  
// 光盘继续转动  
animator.resume();  
  
// 退出应用  
finish();
```

---

## 总 结

本项目为《移动 APP 开发与应用》的课程设计。该项目设计并利用 Android Studio 编程实现了一个小型音乐 App——KK 音乐。该 App 共分为三个界面，分别为登录界面、App 主界面以及音乐播放界面。该 App 实现的主要业务功能有用户的登录与退出、用户数据的储存、歌手查询以及音乐播放等近 10 项功能。

本项目满足了课程设计的所有基本要求。并在此基础上额外实现了包括使用控件 MediaPlayer 实现音乐播放、使用控件 SeekBar 实现音乐进度条的显示及拖拽进度、使用 ObjectAnimator 类实现唱片的旋转与暂停等额外的技术要点，较为完整、高保真地实现了一个基础的音乐 App 从登录、选歌、播放音乐到退出的全流程和动态播放画面的展示。

在此，我想表达对两位认可老师的感谢！非常感谢卢涛老师耐心细致地向我们传授安卓移动 App 开发的基础知识，深入浅出的教学方式也令我对于安卓开发的基本逻辑和 App 的基本理念有了深刻的理解。同时也非常感谢王旭坪老师对于安卓移动 App 开发的进阶知识的讲解，老师的拓展和分享为我在开发这款 App 较为核心的功能时提供了很大的启发。

最后，作为一名物流管理的本科生，在接触此门课程之前从未接触过面向对象的程序设计。因此，本门课程的学习过程中，我补习了 Java 和面向对象程序设计的相关思想。这使得我在掌握本课程所传授知识的同时，也提高了自己对于新的计算机语言的学习能力，锻炼了自己的编程能力。面向对象的诸多思想与物流管理所关注的生产运作过程密切相关，希望在将来能将本门课程所学的知识用到自己所从事的领域中！