

**NANCHANG UNIVERSITY**

**C#程序设计大作业报告**



**题 目:** 感电音乐播放器

**学 院：** 软件学院

**专 业：** 软件工程

**班 级：** 软件工程1809、1810班

**学 号：**  8002118240 、7102118055 、8002118266

**学生姓名：** 杨孟衡 、王敏、何月琦

**完成人数：** 3人

**起讫日期：** 2020-11-28—2021-01-10

**任课教师：** 钟表 职称： 讲师

**完成时间：** 2021年01月08日

**填表日期：** 2021 年 1 月 8 日

**目录**

[1.项目需求 3](#_Toc61094320)

[1.1项目介绍 3](#_Toc61094321)

[1.2需求分析 3](#_Toc61094322)

[1.3项目分工 4](#_Toc61094323)

[2.页面设计 4](#_Toc61094324)

[2.1菜单模块 4](#_Toc61094325)

[2.2播放列表模块 4](#_Toc61094326)

[2.3数据库模块 6](#_Toc61094327)

[3.代码设计 6](#_Toc61094328)

[3.1 工程文件组织结构 6](#_Toc61094329)

[3.2 功能模块 7](#_Toc61094330)

[3.2.1 菜单功能模块 7](#_Toc61094331)

[3.2.2播放列表模块： 7](#_Toc61094332)

[3.2.3 数据库模块 12](#_Toc61094333)

[3.3 代码演示 18](#_Toc61094334)

[4.项目小结 20](#_Toc61094335)

# 1.项目需求

## 1.1项目介绍

随着经济发展迅速，科学技术也在不断发展进步。作为人们生活中重要的娱乐方式——音乐，也随着人们生活的进步，越来越普及起来，人们对音乐的要求也越来越高。并且随着信息技术的进步，人们对音乐的需求日益增长，数字媒体音乐播放器的更新换代愈来愈快。有的播放器功能繁多，有的趋于简约，各式各样的播放器可供人们选择。而一款优质的音乐播放器不仅要求界面美观，而且操作必须简单，这样才能符合大众的要求。本项目的音乐播放器就具备这两点重要的要求，使用一种简单易懂的方法设计一款简单实用的音乐播放器，带你进入一个舒适的音乐享受空间。

## 1.2需求分析

一款简单实用的音乐播放器，也需要满足音乐播放器最基本的功能。在本项目中，通过对用户群体的使用分析，将音乐播放器的功能需求大致分为三部分：

1.菜单功能模块：

播放模式：列表播放、单曲循环、随机播放

搜索功能：从列表中搜索歌曲

歌曲切换与暂停：按钮切换与暂停

播放时间：显示歌曲总时间随着歌曲播放时间轮转

播放速度与声音控制：通过添加Trackbar控件，写入时间，用来控制播放位置，声音控制

专辑照片：将歌曲的专辑照片加入界面中，点击可将照片放大

底盘图标：音乐播放器最小化后会在底部任务栏显示一个图标，右键可以退出程序或将页面最大化

2.播放列表功能模块：

播放列表：显示歌曲信息，双击播放歌曲，右键点击歌曲可选择收藏到最爱歌单，从列表中删除以及打开文件位置

添加文件：添加单个或多个音频文件

删除文件：从播放列表删除指定音频文件

我的最爱列表：将歌曲从播放列表中添加到我的最爱列表中。

双击播放：双击列表中的歌曲进行播放

打开文件类型：MP3 WAV VMA AVI MPG ASF WMV等多种格式的音频文件

3.数据库功能模块：

添加文件到数据库：添加单个或多个音频文件到数据库、添加单首或多首歌曲到收藏记录

从数据库中删除文件：从播放列表删除指定音频文件，从收藏列表中删除指定收藏内容

查询数据库：根据特定要求，查询数据库数据，获取播放列表、收藏列表，获取歌曲信息

获取歌曲信息：根据指定路径，获取音频文件的所有相关信息

## 1.3项目分工

杨孟衡(组长)：菜单功能模块的实现，收集素材，文档整合

何月琦：播放列表功能模块的实现，编写文档

王敏：数据库功能模块的实现，关于页面编写文档

# 2.页面设计

## 2.1菜单模块

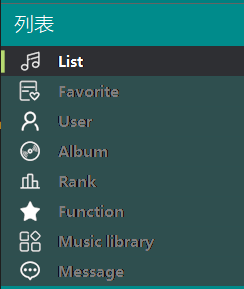
在菜单模块中，整体页面采用的是灰绿色背景设计，左侧菜单项以深色调的黑色为背景色块，菜单图标logo采用的是与深色调有强烈对比差的纯白色调来展示菜单icon样式，整体菜单主要风格为灰绿色风格，搭配播放列表的暗宝石绿色调，可以使得播放器整体风格有亮点但是也不失去统一。

窗口左侧菜单设计内容包括以下几项：

* List:播放列表
* Favorite:收藏列表
* About:关于页面
* Album:专辑
* Rank:播放数据
* Function:功能列表
* Music library:音乐库
* Message:消息窗口

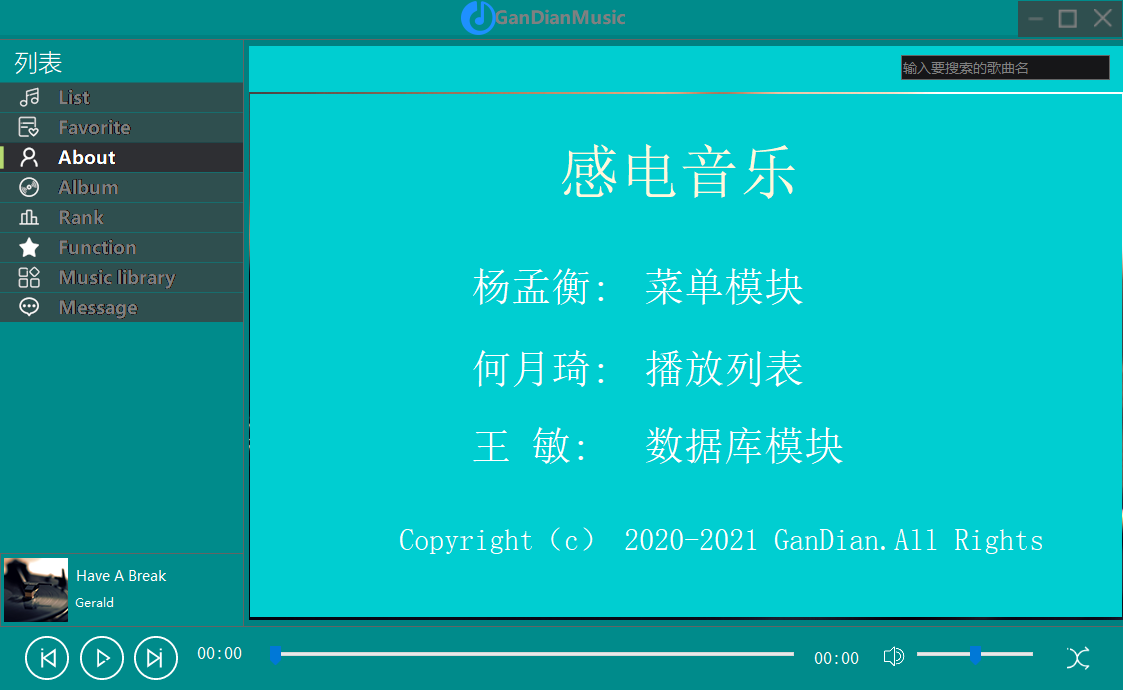
左侧菜单栏应当具有统一的菜单样式，这样风格统一界面会更好看，选择了比较撞色的暗宝石绿与深色调搭配突出菜单项，设计一个选中函数，当某个菜单项获得鼠标焦点时，改变菜单项样式为高亮显示文本，加深背景图层显示，以此来突出选中的菜单项。

设计如下：



本来是打算全部菜单项内容都要实现的，但是后来发现个人能力有限，无法在短时间内实现所有的菜单项内容，所以最后选择了最核心的两个菜单项来实现，然后把这两个菜单项的细节内容做好，实现了Lisi播放列表，Favorite收藏列表两个菜单项内容，然后纵向扩展这两个菜单项设计，详细设计说明放在下一节2.2播放列表模块，本节就不再赘述。

同时，还添加了一个关于菜单页面，主要是展示小组的分工合作信息，嵌套在右侧面板中。



菜单模块部分还要做一些其他的小ui设计，比如程序界面的标题栏，最小化最大化，关闭窗口等功能，以及左侧菜单栏下专辑图片显示，音乐信息显示，整个播放器最重要也是最基础的功能，播放控制的部分，接下来让我们一步一步来解释设计。

* 标题栏设计为播放器logo与播放器名居中显示，当鼠标选中闭包提篮时会高亮显示文本内容，如图：

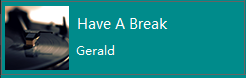


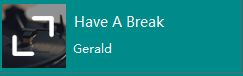


* 最小化最大化关闭按钮设计为当鼠标悬停时，按钮高亮显示来提醒使用者即将选中哪个按钮，如图：



* 设计专辑图片与音乐信息展示区域，将区域分为两大块，左侧一块是专辑图片显示区域，右侧一块是音乐信息，展示歌曲名与歌手名，当鼠标悬停在专辑图片时，高亮显示，提示用户可以点击专辑图片实现专辑图片放大显示，效果如图：

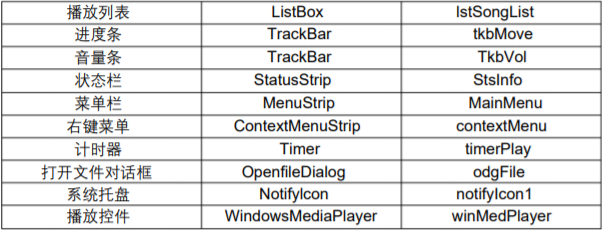
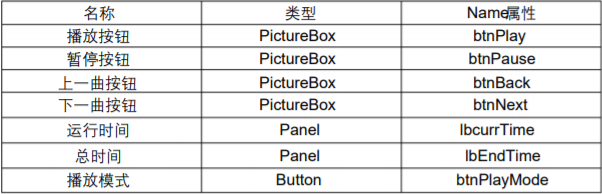




最后一部分是设计播放控制区域，包括上一首，暂停，下一首三个按钮，播放时间label显示，播放时间进度条trackbar，音量进度条trackbar，播放模式按钮，以及一个显示音量的小icon，设计如图：



* 最终展示一下感电音乐页面组件：



## 2.2播放列表模块

在播放列表模块中，页面整体背景采用的是灰绿色设计，主要风格为深色调风格，搭配主菜单的灰绿色与黑色，可以使得播放器整体风格有亮点但是也不失去统一。

在播放列表中，设计的是采用表格的样式，将歌曲的信息展现出来。首先是表头，主要列表项有序号、音乐标题、歌手、专辑、时长以及大小。其中因为觉得把序号这个列的名称展示出来会有点不协调，故将其列表项的名称不进行绘制，但是还是有此列表项。表头采用黑色填充，文字为了不抢眼将其设置为灰色，这样既能够看得清又不会因为太抢眼而不协调。

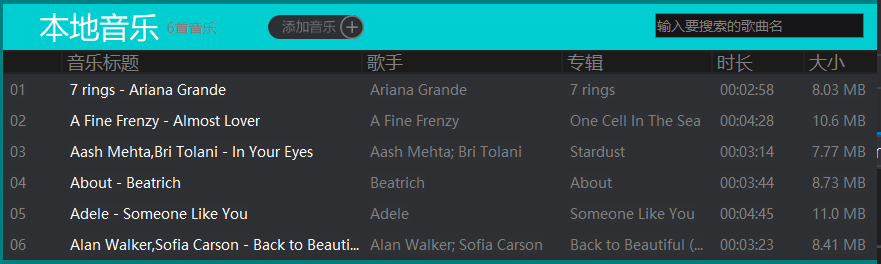
在播放列表项中，将音乐信息逐一根据表头信息绘制出来，其中因为觉得音乐标题是比较重要的信息，故决定将音乐标题的文字绘制时采用白色，从而达到能与主题协调但是却能够亮眼。其他的信息都统一采用灰色的。

将鼠标指向歌曲并点击右键时，会有一个小框，小框中中三个选项。分别是收藏到歌单、从列表中删除以及打开歌曲的文件位置。这三个选项分别都采用的图标与文字的结合。其中收藏到歌单就是将歌曲收藏到我的最爱列表中，故选取的爱心的图标。然后就是从列表中删除，就是将歌曲从播放列表中删除，故采取的是垃圾桶的图标。最后是打开文件位置，就是找到歌曲所处的文件夹并打开文件夹，故选取的文件夹的图标。

在添加歌曲文件中，采用的是小图片的形式，并且有在鼠标没指向此小图标是，图标中的文字是灰色的。在移入之后，就是白色亮起来的。其实就是采用了鼠标移入移出的事件，进行改变图标。

最后在我的最爱列表中样式功能是相似的，不同就是在最爱的列表中鼠标右击歌曲是不会有添加到收藏这一选项了。

* 播放列表设计效果如下：



* 当鼠标指向添加音乐图标时：



* 当鼠标右键点击歌曲时：



在列表中，首先是首行的表头，采用单个函数lvSongList\_DrawColumnHeader进行绘制的表头。在表头绘制函数中，绘制时使用了FillRectangle函数先将表头填充满颜色，然后再调用TextRenderer的DrawText函数在指定的画布中进行绘制文字，也就是表头的列表项文字，例如音乐标题、歌手、专辑等等。然后再新建Pen、Point以及Size类，使用Point和Size类来新建一个矩形，然后再在画布上使用pen类来绘画矩形，达到在一行中分出多个列的效果。

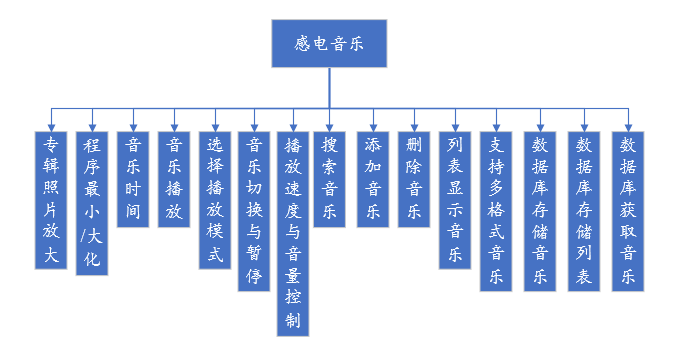
## 2.3数据库模块

数据库模块主要是充当项目与数据库中数据的中介，通过数据库模块中的方法，进行操作数据库内容。本次所实现的音乐播放器的播放，其实是根据数据库中所存储的数据中音乐文件的存放路径，然后根据路径去找到对应的音乐文件，进行歌曲的播放。然后对应音乐收藏，其实也是在数据库中存放了一个专门用于记录收藏歌曲的表格，每收藏一首歌，就往表格中添加一条记录。所以数据库模块就没有具体的界面，但是每个界面都会使用到数据库模块中的方法来完成每一个页面的功能。

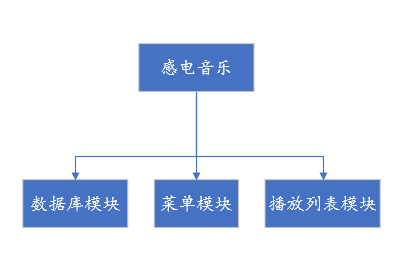
# 3.代码设计

## 3.1 工程文件组织结构

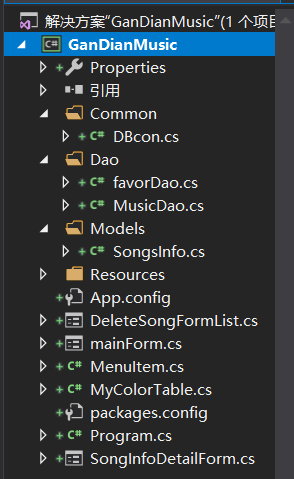
### 3.1.1 系统总功能图



### 3.1.2 系统功能模块图



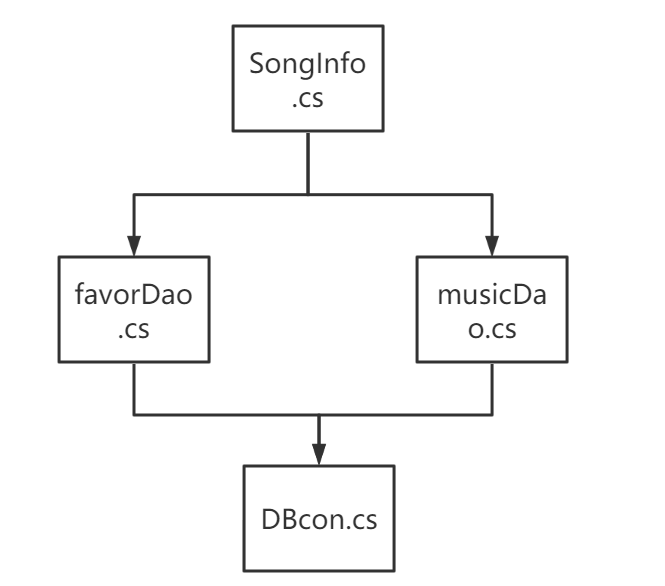
### 3.1.3 项目组织结构



### 3.1.4 项目架构描述

* 根据上一节的模块图，其中菜单模块和播放列表模块并没有太多的架构设计，主要是代码功能实现，而数据库模块则由很详细的架构设计，接下来就详细说明我们项目小组的数据库架构设计。

在数据库模块中，主要将代码分为两个文件，一个是专门实现与数据库连接的文件，同时对数据库进行操作；另一个是将数据库操作的方法进一步封装成类，方便音乐添加等操作。同时也需要在各个面板中定义一些在本面板中需要使用的函数，通过这些函数来实现与数据库的连接。



DBcon.cs主要是实现与数据库进行连接，并实现对数据库中数据进行访问，同时可以对数据进行修改等操作，这是实现音乐播放器的关键所在。而favorDao.cs是DBcon的上一层，是针对收藏歌曲的添加、查询，主要是调用DBcon.cs中的查询和添加方法，实现对收藏歌曲的记录，每当用户添加一首收藏的歌曲，就会先定义一个favorDao对象，通过favorDao对象调用DBcon.cs中的方法，实现添加收藏和显示列表的功能。musicDao.cs同样也是DBcon的上一层，它是针对歌曲本身进行的添加、查询、删除、修改功能，实现这些功能的基础也是调用DBcon.cs中的对应的方法，当用户从文件夹中添加歌曲时，需要获取音乐文件的信息，然后使用MusicDao中的方法添加到数据库中。SongInfo.cs主要是歌曲对象本身的属性和存放的位置等信息，同时包含根据路径信息设置音乐文件本身信息的方法和设置音乐图片的函数。

## 3.2 功能模块

### 3.2.1 菜单功能模块

界面代码设计：

* 标题栏的部分，利用的是VS提供的winform设计界面进行设计，添加了一个label和一个picturebox，以及三个button最小化，最大化，关闭。需要注意的是整个标题栏的背景图层是依靠label控件放大显示，然后居中显示文本内容，让其他控件显示在这个label控件上方，其中鼠标选中高亮函数，以及最小化最大化关闭函数代码设计如：

*/\**

*\* 拖动窗口，标题文字拖动变色（默认gray、拖动white）*

*\*/*

**Point** downPoint;

**private** void lbTitle\_MouseDown(object sender, **MouseEventArgs** e)

        {

            downPoint = new **Point**(e.X, e.Y);

            lbTitle.ForeColor = Color.White;

        }

**private** void lbTitle\_MouseMove(object sender, **MouseEventArgs** e)

        {

            if (e.Button == MouseButtons.Left)

            {

                this.Location = new **Point**(this.Location.X + e.X - downPoint.X, this.Location.Y + e.Y - downPoint.Y);

            }

        }

**private** void lbTitle\_MouseUp(object sender, **MouseEventArgs** e)

        {

            lbTitle.ForeColor = Color.Gray;

        }

*/\**

*\* 关闭、最大化、最小化按钮点击事件*

*\*/*

**private** void FormControlButton\_Click(object sender, **EventArgs** e)

        {

**PictureBox** currPicBox = (**PictureBox**)sender;

            if (currPicBox.Name == "pbCloseForm")

            {

**DialogResult** result = MessageBox.Show("是否退出播放器？", "感电音乐", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

                if (result == DialogResult.Yes)

                    Application.Exit();

            }

            else if (currPicBox.Name == "pbMaxForm")

            {

                this.WindowState = FormWindowState.Maximized;

            }

            else if (currPicBox.Name == "pbMinForm")

            {

                this.WindowState = FormWindowState.Minimized;

            }

        }

* 接下来是设计左侧菜单模块，主要就是设计一个菜单项类，属性包括图标icon，菜单文本内容，然后设置一个菜单栏，把菜单项加入到菜单栏中，以及音乐信息展示部分的设计，专辑图片点击放大显示，音乐信息即时更新显示，要注意的是要在窗体初始化函数里手动添加菜单项，因为菜单项类是我们自创的类，以及部分公共函数处理，比如播放器缩略图处理等，代码设计如：

**public** mainForm()

        {

            InitializeComponent();

*//检测MediaPlayer控件是否有安装*

*//if (testAWM == null)*

*//{*

*//    throw new Exception();*

*//}*

*//else*

*//{*

*//    testAWM.Dispose();*

*//}*

            testAWM.PlayStateChange += new **AxWMPLib**.**\_WMPOCXEvents\_PlayStateChangeEventHandler**(AxWmp\_PlayStateChange);

**MenuItem** item1 = new **MenuItem**(Resources.list, "List");

**MenuItem** item2 = new **MenuItem**(Resources.favorite, "Favorite");

**MenuItem** item3 = new **MenuItem**(Resources.user, "User");

**MenuItem** item4 = new **MenuItem**(Resources.album, "Album");

**MenuItem** item5 = new **MenuItem**(Resources.ranking, "Rank");

**MenuItem** item6 = new **MenuItem**(Resources.star, "Function");

**MenuItem** item7 = new **MenuItem**(Resources.musicLibrary, "Music library");

**MenuItem** item8 = new **MenuItem**(Resources.message, "Message");

            this.menuItemList = new **List**<**MenuItem**>();

            menuItemList.Add(item1);

            menuItemList.Add(item2);

            menuItemList.Add(item3);

            menuItemList.Add(item4);

            menuItemList.Add(item5);

            menuItemList.Add(item6);

            menuItemList.Add(item7);

            menuItemList.Add(item8);

            lbMenu.Items.Add("List");

            lbMenu.Items.Add("Favorite");

            lbMenu.Items.Add("Music library");

            lbMenu.Items.Add("User");

            lbMenu.Items.Add("Album");

            lbMenu.Items.Add("Rank");

            lbMenu.Items.Add("Function");

            lbMenu.Items.Add("Message");

**ImageList** imgList = new **ImageList**();

            imgList.ImageSize = new **Size**(1, 30);    *//分别是宽和高*

            lvSongList.SmallImageList = imgList;

**MyColorTable** myColorTable = new **MyColorTable**();

            cmsSongListMenu.Renderer = new **ToolStripProfessionalRenderer**(myColorTable);

            cmsSongListMenu.ForeColor = Color.White;

            cmsSongListMenu.BackColor = Color.DarkSlateGray;

            pbAddSong.Visible = false;

        }

        #region 窗体Load、Shown、Closed事件

*/\**

*\* 设置各种初始状态*

*\*/*

**private** void Form1\_Load(object sender, **EventArgs** e)

        {

*//设置文件打开窗口（添加音乐）可多选*

            this.odlgFile.Multiselect = true;

*//重置播放器状态信息*

            ReloadStatus();

*//读取播放器列表历史记录*

            localSongsList = ReadHistorySongsList();

            favoriteSongsList = ReadFavorSongsList();

*//设置专辑图片控件到顶部页面（z-index)*

            pbAlbumImage.BringToFront();

*//设置开机自启*

            StarUp("0");

        }

*/\**

*\* 设置任务栏缩略图的属性与绑定事件*

*\*/*

**private** void Form1\_Shown(object sender, **EventArgs** e)

        {

*//暂停按钮*

            ttbbtnPlayPause = new **ThumbnailToolbarButton**(GanDianMusic.Properties.Resources.播放icon, "播放");

            ttbbtnPlayPause.Enabled = true;

            ttbbtnPlayPause.Click += new **EventHandler**<**ThumbnailButtonClickedEventArgs**>(btnPlay\_Click);

*//上一首按钮*

            ttbbtnPre = new **ThumbnailToolbarButton**(GanDianMusic.Properties.Resources.上一首icon, "上一首");

            ttbbtnPre.Enabled = true;

            ttbbtnPre.Click += new **EventHandler**<**ThumbnailButtonClickedEventArgs**>(btnBack\_Click);

*//下一首按钮*

            ttbbtnNext = new **ThumbnailToolbarButton**(GanDianMusic.Properties.Resources.下一首icon, "下一首");

            ttbbtnNext.Enabled = true;

            ttbbtnNext.Click += new **EventHandler**<**ThumbnailButtonClickedEventArgs**>(btnNext\_Click);

            TaskbarManager.Instance.ThumbnailToolbars.AddButtons(this.Handle, ttbbtnPre, ttbbtnPlayPause, ttbbtnNext);

*//裁剪显示略缩图*

*//坐标值为多个父容器相对的位置坐标累加所得*

**Point** p = new **Point**(4, 558);

            TaskbarManager.Instance.TabbedThumbnail.SetThumbnailClip(this.Handle, new **Rectangle**(p, pbSmallAlbum.Size));

        }

*/\**

*\* 窗体关闭事件*

*\*/*

**private** void Form1\_FormClosed(object sender, **FormClosedEventArgs** e)

        {

            Application.Exit();

            this.Dispose();

        }

        #endregion

        #region 窗体左部菜单

*/\**

*\* 绘制单元项*

*\*/*

**private** void lbMenu\_DrawItem(object sender, **DrawItemEventArgs** e)

        {

**Bitmap** bitmap = new **Bitmap**(e.Bounds.Width, e.Bounds.Height);

            int index = e.Index;                                *//获取当前要进行绘制的行的序号，从0开始。*

**Graphics** g = e.Graphics;                            *//获取Graphics对象。*

**Graphics** tempG = Graphics.FromImage(bitmap);

            tempG.SmoothingMode = SmoothingMode.AntiAlias;          *//使绘图质量最高，即消除锯齿*

            tempG.InterpolationMode = InterpolationMode.HighQualityBicubic;

            tempG.CompositingQuality = CompositingQuality.HighQuality;

**Rectangle** bound = e.Bounds;                         *//获取当前要绘制的行的一个矩形范围。*

            string text = this.menuItemList[index].Text.ToString();     *//获取当前要绘制的行的显示文本。*

*//绘制选中时的背景，要注意绘制的顺序，后面的会覆盖前面的*

*//绘制底色*

**Color** backgroundColor = Color.DarkSlateGray;             *//背景色*

**Color** guideTagColor = Color.FromArgb(183, 218, 114);            *//高亮指示色*

**Color** selectedBackgroundColor = Color.FromArgb(46, 47, 51);     *//选中背景色*

**Color** fontColor = Color.Gray;                                   *//字体颜色*

**Color** selectedFontColor = Color.White;                          *//选中字体颜色*

**Font** textFont = new **Font**("微软雅黑", 9, FontStyle.Bold);        *//文字*

*//图标*

**Image** itmeImage = this.menuItemList[index].Img;

*//矩形大小*

**Rectangle** backgroundRect = new **Rectangle**(0, 0, bound.Width, bound.Height);

**Rectangle** guideRect = new **Rectangle**(0, 4, 5, bound.Height - 8);

**Rectangle** textRect = new **Rectangle**(55, 0, bound.Width, bound.Height);

**Rectangle** imgRect = new **Rectangle**(20, 4, 22, bound.Height - 8);

*//当前选中行*

            if ((e.State & DrawItemState.Selected) == DrawItemState.Selected)

            {

                backgroundColor = selectedBackgroundColor;

                fontColor = selectedFontColor;

            }

            else

            {

                guideTagColor = backgroundColor;

            }

*//绘制背景色*

            tempG.FillRectangle(new **SolidBrush**(backgroundColor), backgroundRect);

*//绘制左前高亮指示*

            tempG.FillRectangle(new **SolidBrush**(guideTagColor), guideRect);

*//绘制显示文本*

            TextRenderer.DrawText(tempG, text, textFont, textRect, fontColor,

                                  TextFormatFlags.VerticalCenter | TextFormatFlags.Left);

*//绘制图标*

            tempG.DrawImage(itmeImage, imgRect);

            g.DrawImage(bitmap, bound.X, bound.Y, bitmap.Width, bitmap.Height);

            tempG.Dispose();

        }

*/\**

*\* 设置行高*

*\*/*

**private** void lbMenu\_MeasureItem(object sender, **MeasureItemEventArgs** e)

        {

            e.ItemHeight = 30;

        }

*/\**

*\* 菜单按钮选中事件*

*\*/*

**private** void lbMenu\_SelectedIndexChanged(object sender, **EventArgs** e)

        {

            switch (lbMenu.SelectedIndex)

            {

                case 0:     *//选中的是本地列表*

                    lvSongList.Items.Clear();

                    AddSongsToListView(localSongsList);

                    lvSongList.BringToFront();

                    tsmiFavorite.Visible = true;

                    pbAddSong.Visible = true;

                    lbMusicName.Visible = false;

                    lbCategory.Text = "本地音乐";

                    lbCategory.Visible = true;

                    lbSongCount.Visible = true;

                    break;

                case 1:     *//选中的是收藏音乐*

                    lvSongList.Items.Clear();

                    AddSongsToListView(favoriteSongsList);

                    lvSongList.BringToFront();

                    tsmiFavorite.Visible = false;

                    pbAddSong.Visible = false;

                    lbMusicName.Visible = false;

                    lbCategory.Text = "收藏音乐";

                    lbCategory.Visible = true;

                    lbSongCount.Visible = true;

                    break;

            }

            int songsCount = lvSongList.Items.Count;

            lbSongCount.Text = songsCount + "首音乐";

        }

*/\**

*\* 专辑小图标点击事件*

*\*/*

**private** void pbSmallAlbum\_Click(object sender, **EventArgs** e)

        {

            pbAlbumImage.BringToFront();

            pbAddSong.Visible = false;

            lbSongCount.Visible = false;

            lbCategory.Visible = false;

            lbMusicName.Visible = true;

        }

        #endregion

* 最后是窗体部分的代码设计，包括控制按钮单击事件，音量与进度条事件，其中我们播放器的控件是利用windows系统自带的音乐播放控件AxWindowsMediaPlayer，然后让其在程序中不可见，接着让播放控制区域的功能控件添加鼠标点击事件处理播放功能（下一首，暂停，上一首），同样的原理将音量进度条，播放时间进度条，以及播放时间label链接到我们自创的功能性控件上，最后还需要多加一个计时器timer，用来同步我们自创的进度条，同步播放进度，代码如：

 #region 播放核心

*/\**

*\* 播放指定的歌曲*

*\* index：播放列表中，歌曲的序号*

*\*/*

**private** void Play(int index)

        {

*//设置被播放音乐项的状态*

            lvSongList.Items[index].Focused = true;

            lvSongList.Items[index].EnsureVisible();

            lvSongList.Items[index].Selected = true;

*//Console.WriteLine(lvSongList.Items[index].SubItems[6].Text);*

            if (AxWmp.playState.ToString() == "wmppsPlaying")       *//播放->其他状态*

            {

                AxWmp.Ctlcontrols.pause();

                pbPlay.BackgroundImage = Resources.播放hoover;

                ttbbtnPlayPause.Icon = Resources.播放icon;

                return;

            }

            else if (AxWmp.playState.ToString() != "wmppsPaused")      *//更改播放路径并播放*

            {

*//生成随机序列*

                BuildRandomList(lvSongList.Items.Count);

                jumpSongIndex = index;

                currPlaySong = new **SongsInfo**(lvSongList.SelectedItems[0].SubItems[6].Text);

                AxWmp.URL = currPlaySong.FilePath;

                AxWmp.Ctlcontrols.play();

                return;

            }

            else                            *//暂停->播放*

                AxWmp.Ctlcontrols.play();

            pbPlay.BackgroundImage = Resources.暂停hoover;

            ttbbtnPlayPause.Icon = Resources.暂停icon;

            ttbbtnPlayPause.Tooltip = "暂停";

        }

*/\**

*\* 计时器函数*

*\*/*

**private** void timerPlay\_Tick(object sender, **EventArgs** e)

        {

*//设置当前播放时间*

            lbcurrTime.Text = AxWmp.Ctlcontrols.currentPositionString;

            lbEndTime.Text = currPlaySong.Duration.Remove(0, 3);

*//设置滑动条值*

            tkbMove.Value = (int)AxWmp.Ctlcontrols.currentPosition;

        }

*/\**

*\* 播放器控件状态改变事件*

*\*/*

**private** void AxWmp\_PlayStateChange(object sender, **AxWMPLib**.**\_WMPOCXEvents\_PlayStateChangeEvent** e)

        {

            switch (e.newState)

            {

                case 0:    *// Stopped 未知状态*

                    break;

                case 1:    *// Stopped 停止*

                    timerPlay.Stop();

                    ReloadStatus();

                    break;

                case 2:    *// Paused 暂停*

                    timerPlay.Stop();

                    break;

                case 3:    *// Playing 正在播放*

                    timerPlay.Start();

*//显示专辑图片*

                    pbAlbumImage.Image = currPlaySong.AlbumImage;

                    pbSmallAlbum.BackgroundImage = currPlaySong.SmallAblum;

                    lbSmallAlbumSingerName.Text = currPlaySong.Artist;

*//显示歌曲标题名字*

                    lbMusicName.Text = currPlaySong.FileName;

                    if (currPlaySong.FileName.Length > 18)

                        lbSmallAlbumSongName.Text = currPlaySong.FileName.Substring(0, 18) + "...";

                    else

                        lbSmallAlbumSongName.Text = currPlaySong.FileName;

                    tkbMove.Maximum = (int)AxWmp.currentMedia.duration;

                    int currIndex = lvSongList.SelectedItems[0].Index;

                    lvSongList.SelectedItems.Clear();

                    lvSongList.Items[currIndex].Selected = true;    *//设定选中*

                    lvSongList.Items[currIndex].EnsureVisible();    *//保证可见*

                    lvSongList.Items[currIndex].Focused = true;

                    lvSongList.Select();

                    break;

                case 4:    *// ScanForward*

*//tsslCurrentPlayState.Text = "ScanForward";*

                    break;

                case 5:    *// ScanReverse*

*//tsslCurrentPlayState.Text = "ScanReverse";*

                    break;

                case 6:    *// Buffering*

*//tsslCurrentPlayState.Text = "正在缓冲";*

                    break;

                case 7:    *// Waiting*

*//tsslCurrentPlayState.Text = "Waiting";*

                    break;

                case 8:    *// MediaEnded*

*//tsslCurrentPlayState.Text = "MediaEnded";*

                    break;

                case 9:    *// Transitioning*

*//tsslCurrentPlayState.Text = "正在连接";*

                    break;

                case 10:   *// Ready*

*//tsslCurrentPlayState.Text = "准备就绪";*

                    break;

                case 11:   *// Reconnecting*

*//tsslCurrentPlayState.Text = "Reconnecting";*

                    break;

                case 12:   *// Last*

*//tsslCurrentPlayState.Text = "Last";*

                    break;

                default:

*//tsslCurrentPlayState.Text = ("Unknown State: " + e.newState.ToString());*

                    break;

            }

            if (AxWmp.playState.ToString() == "wmppsMediaEnded")

            {

*//Console.WriteLine(lvSongList.SelectedItems[0].Index + ":播放完毕");*

*//获取音乐播放文件路径，并添加到播放控件*

                string path = GetPath();

**WMPLib**.**IWMPMedia** media = AxWmp.newMedia(path);

                AxWmp.currentPlaylist.appendItem(media);

            }

        }

*/\**

*\* 重置播放器状态信息*

*\*/*

**private** void ReloadStatus()

        {

*//设置专辑封面为默认*

            pbAlbumImage.Image = GanDianMusic.Properties.Resources.defaultAlbum;

            lbMusicName.Text = "GanDianMusic";

            lbcurrTime.Text = "00:00";

            lbEndTime.Text = "00:00";

            tkbVol.Value = tkbVol.Maximum / 2;

            tkbMove.Value = 0;

            if (lvSongList.Items.Count > 0 && lvSongList.SelectedItems.Count == 0)

            {

                lvSongList.Items[0].Selected = true;*//设定选中*

                lvSongList.Items[0].EnsureVisible();*//保证可见*

                lvSongList.Items[0].Focused = true;

            }

        }

        #endregion

         #region 窗体底部

        #region 控制按钮单击事件

**private** void btnPlay\_Click(object sender, **EventArgs** e)

        {

            if (AxWmp.playState.ToString() == "wmppsPlaying")       *//播放->暂停*

            {

                AxWmp.Ctlcontrols.pause();

                pbPlay.BackgroundImage = Resources.播放hoover;

                ttbbtnPlayPause.Icon = Resources.播放icon;

                return;

            }

            else if (AxWmp.playState.ToString() == "wmppsPaused")    *//暂停->播放*

            {

                AxWmp.Ctlcontrols.play();

                pbPlay.BackgroundImage = Resources.暂停hoover;

                ttbbtnPlayPause.Icon = Resources.暂停icon;

                return;

            }

            if (lvSongList.SelectedItems.Count > 0)         *//双击播放列表控制*

            {

                Play(lvSongList.SelectedItems[0].Index);

            }

            else

                MessageBox.Show("请选择要播放的曲目");

        }

**private** void btnBack\_Click(object sender, **EventArgs** e)

        {

            if (lvSongList.Items.Count == 0)

            {

                MessageBox.Show("请先添加曲目至目录");

                return;

            }

            int currIndex = lvSongList.SelectedItems[0].Index;

            if (currIndex > 0)

            {

                AxWmp.Ctlcontrols.stop();

                currIndex -= 1;

            }

            else

            {

                AxWmp.Ctlcontrols.stop();

                currIndex = lvSongList.Items.Count - 1;

            }

            lvSongList.Items[currIndex].Focused = true;

            lvSongList.Items[currIndex].EnsureVisible();

            lvSongList.Items[currIndex].Selected = true;

            Play(currIndex);

        }

**private** void btnNext\_Click(object sender, **EventArgs** e)

        {

            if (lvSongList.SelectedItems.Count == 0)

            {

                MessageBox.Show("请先添加曲目至目录");

                return;

            }

            int currIndex = lvSongList.SelectedItems[0].Index;

            if (currIndex < lvSongList.Items.Count - 1)

            {

                AxWmp.Ctlcontrols.stop();

                currIndex += 1;

            }

            else

            {

                AxWmp.Ctlcontrols.stop();

                currIndex = 0;

            }

            Play(currIndex);

        }

**private** void btnPlayMode\_Click(object sender, **EventArgs** e)

        {

            if (currPlayMode == PlayMode.ListLoop)

                currPlayMode = PlayMode.Shuffle;

            else

                currPlayMode += 1;

            if (currPlayMode == PlayMode.SingleLoop)

                btnPlayMode.BackgroundImage = GanDianMusic.Properties.Resources.单曲循环;

            else if (currPlayMode == PlayMode.ListLoop)

                btnPlayMode.BackgroundImage = GanDianMusic.Properties.Resources.列表循环;

            else

                btnPlayMode.BackgroundImage = GanDianMusic.Properties.Resources.随机播放;

        }

        #endregion

        #region 音量与进度条事件

**private** void tkbVol\_ValueChanged(object sender, **EventArgs** e)

        {

            AxWmp.settings.volume = tkbVol.Value;

*//lbVolumeVal.Text = tkbVol.Value.ToString() + "%";*

        }

**private** void tkbVol\_MouseHover(object sender, **EventArgs** e)

        {

*//lbVolumeVal.Text = tkbVol.Value.ToString() + "%";*

        }

**private** void tkbVol\_MouseLeave(object sender, **EventArgs** e)

        {

*//lbVolumeVal.Text = "音量：";*

        }

**private** void tkbMove\_Scroll(object sender, **EventArgs** e)

        {

            AxWmp.Ctlcontrols.currentPosition = (double)this.tkbMove.Value;

        }

        #endregion

*/\**

*\* 系统托盘菜单——退出*

*\*/*

**private** void tsmiQuit\_Click(object sender, **EventArgs** e)

        {

            notifyIcon1.Visible = false;

            notifyIcon1.Dispose();

            this.Close();

        }

        #endregion

#region 系统托盘

*/\**

*\* 窗体大小改变事件*

*\*/*

**private** void Form1\_SizeChanged(object sender, **EventArgs** e)

        {

            if (this.WindowState == FormWindowState.Minimized)

            {

                this.Visible = false;

                notifyIcon1.Visible = true;

            }

        }

*/\**

*\* 系统托盘双击事件*

*\*/*

**private** void notifyIcon1\_DoubleClick(object sender, **EventArgs** e)

        {

            this.Visible = true;

            notifyIcon1.Visible = false;

            this.WindowState = FormWindowState.Normal;

        }

*/\**

*\* 系统托盘菜单——打开*

*\*/*

**private** void tsmiOpenForm\_Click(object sender, **EventArgs** e)

        {

            this.Visible = true;

            notifyIcon1.Visible = false;

            this.WindowState = FormWindowState.Normal;

        }

* 这一部分是播放器的控制模式设计，主要是随机播放，循环播放，单曲循环的实现，循环播放，单曲循环的实现简单，只需要遍历整个播放列表就可以实现循环播放，单曲循环只需要固定播放一首歌即可，而较为困难的是随机播放，我们需要生成一个随机播放序列，保证每一轮播放每一首歌只会播放一次，代码如：

        #region 播放模式

**private** string GetPath()

        {

            int currIndex = lvSongList.SelectedItems[0].Index;

            switch (currPlayMode)

            {

                case PlayMode.ListLoop:

                    if (currIndex != lvSongList.Items.Count - 1)

                        currIndex += 1;

                    else

                        currIndex = 0;

                    break;

                case PlayMode.SingleLoop:

                    Console.WriteLine("SingleLoop");

*//do nothing*

                    break;

                case PlayMode.Shuffle:

*//当局结束*

                    if (randomListIndex > randomList.Length - 1)

                        StarNewRound();

*//匹配到需要跳过的歌曲*

                    if (randomList[randomListIndex] == jumpSongIndex)

                        if (randomListIndex == randomList.Length - 1)   *//当局结束*

                            StarNewRound();

                        else

                            randomListIndex++;

                    currIndex = randomList[randomListIndex++];

                    break;

            }

            lvSongList.Items[currIndex].Selected = true;*//设定选中*

            lvSongList.Items[currIndex].EnsureVisible();*//保证可见*

            lvSongList.Items[currIndex].Focused = true;

            currPlaySong = new **SongsInfo**(lvSongList.SelectedItems[0].SubItems[6].Text);

            return currPlaySong.FilePath;

        }

**private** void StarNewRound()

        {

*//重新生成随机序列*

            BuildRandomList(lvSongList.Items.Count);

*//第二轮开始 播放所有歌曲 不跳过*

            jumpSongIndex = -1;

        }

**private** void BuildRandomList(int songListCount)

        {

            randomListIndex = 0;

            randomList = new int[songListCount];

*//初始化序列*

            for (int i = 0; i < songListCount; i++)

            {

                randomList[i] = i;

            }

*//随机序列*

            for (int i = songListCount - 1; i >= 0; i--)

            {

**Random** r = new **Random**(Guid.NewGuid().GetHashCode());

                int j = r.Next(0, songListCount - 1);

                swap(randomList, i, j);

            }

*//输出序列*

*//for (int i = 0; i < songListCount; i++)*

*//{*

*//    Console.Write(randomList[i] + " ");*

*//}*

*//Console.WriteLine(" ");*

        }

**private** void swap(int[] arr, int a, int b)

        {

            int temp = arr[a];

            arr[a] = arr[b];

            arr[b] = temp;

        }

        #endregion

### 3.2.2播放列表模块：

界面代码设计：

在列表头中，首行的表头采用单个函数lvSongList\_DrawColumnHeader进行绘制的表头。在表头绘制函数中，绘制时使用了FillRectangle函数先将表头填充满颜色，然后再调用TextRenderer的DrawText函数在指定的画布中进行绘制文字，也就是表头的列表项文字，例如音乐标题、歌手、专辑等等。然后再新建Pen、Point以及Size类，使用Point和Size类来新建一个矩形，然后再在画布上使用pen类来绘画矩形，达到在一行中分出多个列的效果。

在列表项中，采用在lvSongList\_DrawSubItem函数中进行绘制列表项。在此函数中，是一项一项绘制出来的。如果当前绘制项的序号是偶数，则进行将序号列的背景绘制成黑色，这样有一种交替的感觉。判断当前绘制的列是否为第一个，也就是音乐标题，如果是，则将其字体颜色绘制成白色，否则别的都是灰色。同理使用填充矩形绘制行。并且调用自己所创建的绘制文字函数DrawText函数进行绘制列表项的内容。重新写一个DrawText的目的是希望能够绘制出指定的对其方式以及在重绘时文本向右移，从而达到整齐的效果。

* 页面绘制主要代码设计：

*//列表头*

        private **void** lvSongList\_DrawColumnHeader(object sender, DrawListViewColumnHeaderEventArgs e)

        {

**int** index = e.ColumnIndex;

            e.Graphics.FillRectangle(new SolidBrush(Color.FromArgb(27, 27, 25)), e.Bounds);

            TextRenderer.DrawText(e.Graphics, lvSongList.Columns[index].Text, new Font("微软雅黑", 9, FontStyle.Regular), e.Bounds, Color.Gray, TextFormatFlags.VerticalCenter | TextFormatFlags.Left);

            Pen pen = new Pen(Color.FromArgb(34, 35, 39), 2);

            Point p = new Point(e.Bounds.Left - 1, e.Bounds.Top + 1);

            Size s = new Size(e.Bounds.Width, e.Bounds.Height - 2);

            Rectangle r = new Rectangle(p, s);

            e.Graphics.DrawRectangle(pen, r);

        }

*//列表项*

        private **void** lvSongList\_DrawSubItem(object sender, DrawListViewSubItemEventArgs e)

        {

            Console.WriteLine("selectedCount:" + lvSongList.SelectedItems.Count);

            if (e.ItemIndex == -1)

            {

                return;

            }

            if (e.ItemIndex % 2 == 0)

            {

                e.SubItem.BackColor = Color.FromArgb(27, 29, 32);

                e.DrawBackground();

            }

            if (e.ColumnIndex == 1)

            {

                e.SubItem.ForeColor = Color.White;

            }

            else

            {

                e.SubItem.ForeColor = Color.Gray;

            }

            if ((e.ItemState & ListViewItemStates.Selected) == ListViewItemStates.Selected)

            {

                using (SolidBrush brush = new SolidBrush(Color.Blue))

                {

                    e.Graphics.FillRectangle(new SolidBrush(Color.FromArgb(46, 47, 51)), e.Bounds);

                }

            }

            if (!string.IsNullOrEmpty(e.SubItem.Text))

            {

                this.DrawText(e, e.Graphics, e.Bounds, 2);

            }

        }

*//列表内容*

        private **void** DrawText(DrawListViewSubItemEventArgs e, Graphics g, Rectangle r, **int** paddingLeft)

        {

            TextFormatFlags flags = GetFormatFlags(e.Header.TextAlign);

            r.X += 1 + paddingLeft;*//重绘图标时，文本右移*

            TextRenderer.DrawText(

                g,

                e.SubItem.Text,

                e.SubItem.Font,

                r,

                e.SubItem.ForeColor,

                flags);

        }

* 功能代码设计：

添加音乐文件功能：先初始化打开文件对话框，然后设置可以被选中的文件格式。如果对话框显示成功，则将进行循环遍历所选中的歌曲。先获取文件路径，判断当前文件路径是否存在，如果存在，则可以进行数据库存入处理。新建一个数据库Dao对象，再新建歌曲信息对象，分别进行赋值。然后调取MusicDao对象的Addmusic函数将歌曲写入数据库，同理也将歌曲写入本地歌曲列表中。最后将本地歌曲列表通过调用函数AddSongsToListView加载到播放列表中进行显示。最后进行保存歌曲列表历史，然后再更新到播放列表源中。

删除音乐文件功能：先将当前被选中的歌曲新建一个被删除的歌曲列表对象，然后显示是否确认删除当前歌曲。如果是，则获取当前的列表是本地播放列表还是我的最爱列表。分别调用其列表的RemoveAt，来进行删除指定序号的歌曲。再将删除过后的歌曲列表重新保存进保存列表的历史纪录。最后才将删除后的歌曲列表重新加入列表显示，然后更新播放来源列表。

双击音乐进行播放功能：首先获取当前被选中的序号以及歌曲路径，进行判断当前正在播放的是否为被选中的歌曲，如果是，则再进行判断当前被选中时是否有正在播放，如果没有，则进行播放；如果有，则进行暂停播放。如果当前选择的歌曲不是目前正在播放的歌曲，则将先将列表进行随机排序，相当于生成了随机播放的列表，然后再将目前所选的序号赋值给要跳转到歌曲的序号，再将目前所播放歌曲重新赋值为当前被选中的歌曲，最后再使用AxWmp来进行播放。

* 主要代码设计：

        #region 添加音乐

        private **void** tsmiOpenFile\_Click(object sender, EventArgs e)

        {

            this.odlgFile.InitialDirectory = defaultSongsFilePath;

            this.odlgFile.Filter = "媒体文件|\*.mp3;\*.wav;\*.wma;\*.avi;\*.mpg;\*.asf;\*.wmv";

            if (odlgFile.ShowDialog() == DialogResult.OK)

            {

                for (**int** i = 0; i < odlgFile.FileNames.Length; i++)

                {

                    string path = odlgFile.FileNames[i];

                    if (!IsExistInList(path))

                    {

                        MusicDao musicDao = new MusicDao();

                        SongsInfo song = new SongsInfo(path);

                        MusicDao.Addmusic(song);

                        localSongsList.Add(song);

                    }

                }

            }

            AddSongsToListView(localSongsList);

            SaveSongsListHistory("music",localSongsList);

            UpdataOringinSongList();

        }

*/\**

*\* 菜单——删除音乐*

*\*/*

        private **void** tsmiRemoveSongFromList\_Click(object sender, EventArgs e)

        {

            DeleteSongFormList deleteSongFormList = new DeleteSongFormList(currSelectedSong.FilePath);

            if (deleteSongFormList.ShowDialog() == DialogResult.OK)

            {

**int** removeIndex = lvSongList.SelectedItems[0].Index;

                if (lbMenu.SelectedIndex == 0)

                {

                    localSongsList.RemoveAt(removeIndex);

                    SaveSongsListHistory("music", localSongsList);

                    AddSongsToListView(localSongsList);

                }

                else if (lbMenu.SelectedIndex == 1)

                {

                    favoriteSongsList.RemoveAt(removeIndex);

                    SaveSongsListHistory("favor", favoriteSongsList);

                    AddSongsToListView(favoriteSongsList);

                }

                UpdataOringinSongList();

            }

        }

*/\**

*\* 播放列表双击事件*

*\*/*

        private **void** lvSongList\_DoubleClick(object sender, EventArgs e)

        {

            Console.WriteLine(lvSongList.SelectedItems[0].Index);

**int** currIndex = lvSongList.SelectedItems[0].Index;

            string songFilePath = lvSongList.Items[currIndex].SubItems[6].Text;

            if (currPlaySong.FilePath == songFilePath)

            {

*//选中歌曲为正在播放的歌曲*

                if (AxWmp.playState.ToString() == "wmppsPlaying")

                {

                    AxWmp.Ctlcontrols.pause();

                    pbPlay.BackgroundImage = Resources.播放;

                    ttbbtnPlayPause.Icon = Resources.播放icon;

                }

                else if (AxWmp.playState.ToString() == "wmppsPaused")

                {

                    AxWmp.Ctlcontrols.play();

                    pbPlay.BackgroundImage = Resources.暂停;

                    ttbbtnPlayPause.Icon = Resources.暂停icon;

                }

            }

            else

            {

*//选中的为其他歌曲*

                BuildRandomList(lvSongList.Items.Count);

                jumpSongIndex = currIndex;

                currPlaySong = new SongsInfo(songFilePath);

                AxWmp.URL = songFilePath;

                AxWmp.Ctlcontrols.play();

                pbPlay.BackgroundImage = Resources.暂停;

                ttbbtnPlayPause.Icon = Resources.暂停icon;

            }

            lvSongList.Items[currIndex].Focused = true;

        }

### 3.2.3 数据库模块

（1）DBcon.cs实现与数据库的连接。使用MySQLConnection与数据库建立连接。针对不同的操作定义不同的方法。定义方法NonQuery（），主要实现对数据库执行非查询操作，也就是执行添加、修改、删除操作。定义方法Query（）实现对数据库的查询操作。

* NonQuery代码设计：

        private **static** string constr = "server=127.0.0.1;port=3306;database=musicdb;user=root;password=123456;";

        private **static** MySqlConnection conn = new MySqlConnection(constr);

        public **static** **int** NonQuery(string sql, MySqlParameter[] sp)

        {

**int** value=-1;

            MySqlTransaction st = null;

            try

            {

                conn.Open();

*//开启事务*

                st = conn.BeginTransaction();

*//创建对象*

                MySqlCommand cmd = new MySqlCommand(sql, conn, st);

                if(sp!=null&&sp.Length!=0)

                {

                    cmd.Parameters.AddRange(sp);

                }

*//执行对表的非查询操作*

                value = cmd.ExecuteNonQuery();

                st.Commit();

            }

            catch (Exception ex)

            {

                MessageBox.Show(ex.Message);

*//回滚事务*

                if (st != null)

                {

                    st.Rollback();

                }

            }

            finally

            {

                if (conn.State == ConnectionState.Open)

                {

                    conn.Close();

                }

            }

            return value;

        }

* Query函数代码设计：

        public **static** DataSet Query(string sql, MySqlParameter[] sp)

        {

            DataSet ds = null;

            try

            {

                conn.Open();

                MySqlCommand cmd = new MySqlCommand(sql, conn);

                if(sp!=null&&sp.Length!=0)

                {

                    cmd.Parameters.AddRange(sp);

                }

                MySqlDataAdapter sda = new MySqlDataAdapter(cmd);

                ds = new DataSet();

                sda.Fill(ds);

            }

            catch(Exception ex)

            {

                MessageBox.Show(ex.Message);

            }

            finally

            {

                if(conn.State == ConnectionState.Open)

                {

                    conn.Close();

                }

            }

            return ds;

        }

（2）favorDao.cs实现收藏歌曲。定义方法Addmusic（），主要实现添加多首收藏歌曲，定义方法QueryAll（），实现对数据库中收藏歌曲的查询，定义方法AddFavor（），主要实现的是根据将当前歌曲添加为收藏歌曲，定义方法ClearFavor（），实现删除收藏的歌曲。

* 歌曲收藏代码设计原理上相差无几，都需要先定义一条SQL语句，如果是对多首歌曲添加收藏，则遍历传入的集合，依次添加，如果是单首歌曲，则只需要直接添加即可，都通过调用DBcon.cs中的非查询方法，代码主要设计如：

            string sql = "insert into favor(filename, filepath, filesize, artist, album, albumImage, year, originname, duration, byterate, smallalbum) values"

                + "(" + "@fn, @fp, @fs, @art, @alb, @albI, @year, @ogn, @dur, @br, @sba)";

            MySqlParameter[] sp = new MySqlParameter[11];

            sp[0] = new MySqlParameter("fn", favor.FileName);

            sp[1] = new MySqlParameter("fp", favor.FilePath);

            sp[2] = new MySqlParameter("fs", favor.Filesize);

            sp[3] = new MySqlParameter("art", favor.Artist);

            sp[4] = new MySqlParameter("alb", favor.Album);

            sp[5] = new MySqlParameter("albI", favor.AlbumImage);

            sp[6] = new MySqlParameter("year", favor.Year);

            sp[7] = new MySqlParameter("ogn", favor.OriginName);

            sp[8] = new MySqlParameter("dur", favor.Duration);

            sp[9] = new MySqlParameter("br", favor.ByteRate);

            sp[10] = new MySqlParameter("sba", favor.SmallAblum);

            value = DBcon.NonQuery(sql, sp);

            string sql = "select \* from favor";

            DataSet ds = DBcon.Query(sql, null);

（3）musicDao.cs实现音乐播放。定义方法Addmusic（），实现对一首歌曲的添加，定义方法QueryByName（），实现根据歌曲名查询歌曲，定义方法QueryByArtist（），实现根据歌手查询歌曲，定义方法QueryAllMusic（）实现查询所有歌曲，可以显示播放列表，定义方法DeleteByName()，可以根据当前歌曲名删除数据库中歌曲的信息，从而删除该歌曲，定义方法AddMusic（），实现对多首音乐同时添加的功能。

* 实现对歌曲的添加。歌曲的添加同样也是分为对多首歌曲的添加和对单首歌曲的添加，当需要添加多首歌曲时，就需要遍历集合，依次实现对歌曲的添加，而添加单首歌曲则直接调用DBcon中的方法添加即可，代码设计如：

            string sql = "insert into music(filename, filepath, filesize, artist, album, albumImage, year, originname, duration, byterate, smallalbum) values" +

                "(" + "@fn, @fp, @fs, @art, @alb, @albI, @year, @ogn, @dur, @br, @sba)";

            foreach (SongsInfo music in list)

            {

                MySqlParameter[] sp = new MySqlParameter[11];

                sp[0] = new MySqlParameter("fn", music.FileName);

                sp[1] = new MySqlParameter("fp", music.FilePath);

                sp[2] = new MySqlParameter("fs", music.Filesize);

                sp[3] = new MySqlParameter("art", music.Artist);

                sp[4] = new MySqlParameter("alb", music.Album);

                sp[5] = new MySqlParameter("albI", music.AlbumImage);

                sp[6] = new MySqlParameter("year", music.Year);

                sp[7] = new MySqlParameter("ogn", music.OriginName);

                sp[8] = new MySqlParameter("dur", music.Duration);

                sp[9] = new MySqlParameter("br", music.ByteRate);

                sp[10] = new MySqlParameter("sba", music.SmallAblum);

                value = DBcon.NonQuery(sql, sp);

            }

（4）实现根据路径信息设置音乐文件本身的属性。定义音乐类，包含文件名、文件路径、文件大小等属性，定义方法SongInfo（），设置音乐的文件的路径和音乐图片，定义方法SetSongInfo（），根据路径获取音乐信息并为属性赋值，同时定义方法SetAllbumArt（），实现为音乐文件设置音乐图片，同时定义函数Bitmap实现位图缓冲，将音乐图片先绘制在位图上，再绘制到面板上。

* 实现根据路径获取歌曲信息并设置歌曲属性。首先歌曲本身需要设置较多的属性来记录这些信息，在SetSongInfo方法中，首先根据已知路径依次去获取文件名、文件位置、文件大小等信息，并将这些信息赋值给歌曲对象，代码设计如：

                ShellClass sh = new ShellClass();

                Folder dir = sh.NameSpace(Path.GetDirectoryName(strPath));

                FolderItem item = dir.ParseName(Path.GetFileName(strPath));

                fileName = dir.GetDetailsOf(item, 0);

                fileName = dir.GetDetailsOf(item, 0).Split('.')[0];

                if (fileName == string.Empty)

                    fileName = "未知";

                FilePath = strPath;

                filesize = dir.GetDetailsOf(item, 1);

                if (filesize == string.Empty)

                    filesize = "未知";

                artist = dir.GetDetailsOf(item, 13);

                if (artist == string.Empty)

                    artist = "未知";

                album = dir.GetDetailsOf(item, 14);

                if (album == string.Empty)

                    album = "未知";

                year = dir.GetDetailsOf(item, 15);

                if (year == string.Empty)

                    year = "未知";

                OriginName = dir.GetDetailsOf(item, 21);

                if (OriginName == string.Empty)

                    OriginName = "未知";

                duration = dir.GetDetailsOf(item, 27);

                if (duration == string.Empty)

                    duration = "未知";

* 实现歌曲图片的设置。根据路径信息，获取图片的信息，当图片不为空时，调用Bitmap方法，实现将图片加载到指定大小的位图中，代码设计如：

            if(strPath != "" && strPath != null)

            {

                TagLib.File file = TagLib.File.Create(strPath);

                if (file.Tag.Pictures.Length > 0)

                {

                    var bin = (byte[])(file.Tag.Pictures[0].Data.Data);

                    albumImage = Image.FromStream(new MemoryStream(bin)).GetThumbnailImage(900, 900, null, IntPtr.Zero);

                    albumImage = Cut((Bitmap)albumImage , 20, 215, 877, 530);

*//albumImage = Image.FromStream(new MemoryStream(bin)).GetThumbnailImage(640, 360, null, IntPtr.Zero);*

                    smallAblum = Image.FromStream(new MemoryStream(bin)).GetThumbnailImage(64, 64, null, IntPtr.Zero);

                    return;

                }

            }

* 实现根据路径获取歌曲信息并设置歌曲属性。首先歌曲本身需要设置较多的属性来记录这些信息，在SetSongInfo方法中，首先根据已知路径依次去获取文件名、文件位置、文件大小等信息，并将这些信息赋值给歌曲对象。实现歌曲图片的设置。根据路径信息，获取图片的信息，当图片不为空时，调用Bitmap方法，实现将图片加载到指定大小的位图中，代码设计如：

                ShellClass sh = new ShellClass();

                Folder dir = sh.NameSpace(Path.GetDirectoryName(strPath));

                FolderItem item = dir.ParseName(Path.GetFileName(strPath));

                fileName = dir.GetDetailsOf(item, 0);

                fileName = dir.GetDetailsOf(item, 0).Split('.')[0];

                if (fileName == string.Empty)

                    fileName = "未知";

                FilePath = strPath;

                filesize = dir.GetDetailsOf(item, 1);

                if (filesize == string.Empty)

                    filesize = "未知";

                artist = dir.GetDetailsOf(item, 13);

                if (artist == string.Empty)

                    artist = "未知";

                album = dir.GetDetailsOf(item, 14);

                if (album == string.Empty)

                    album = "未知";

                year = dir.GetDetailsOf(item, 15);

                if (year == string.Empty)

                    year = "未知";

                OriginName = dir.GetDetailsOf(item, 21);

                if (OriginName == string.Empty)

                    OriginName = "未知";

                duration = dir.GetDetailsOf(item, 27);

                if (duration == string.Empty)

                    duration = "未知";

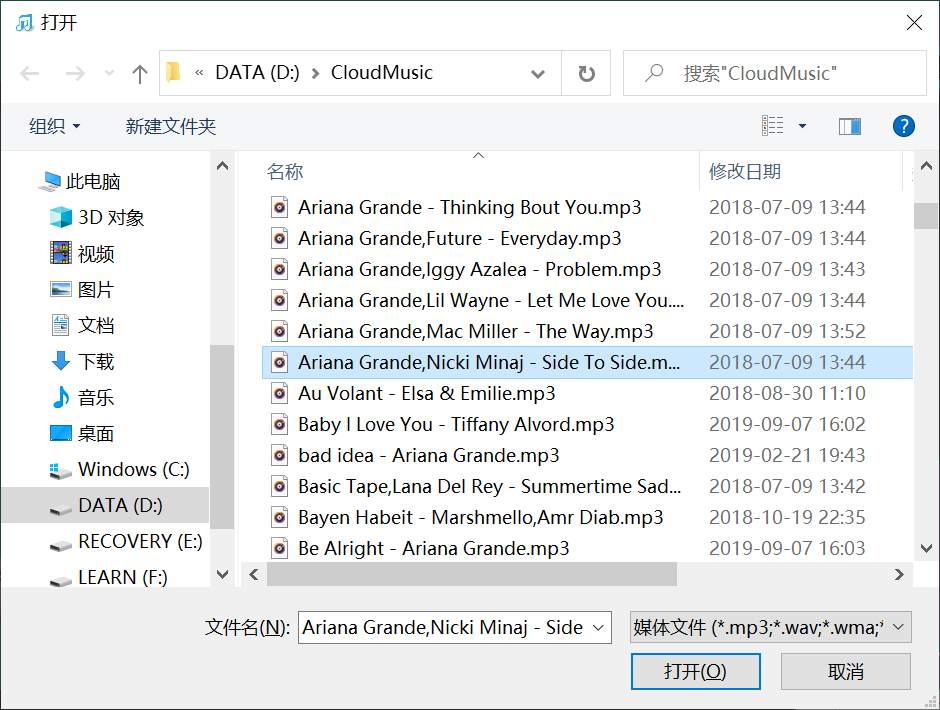
## 3.3 代码演示

* 双击播放音乐：



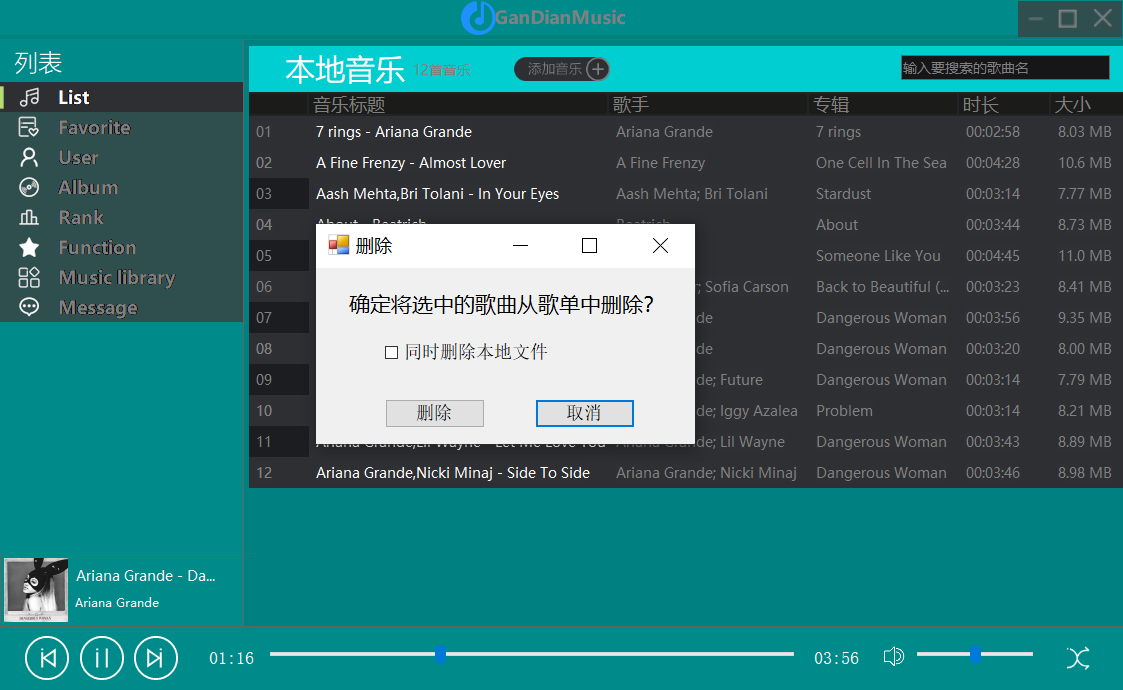
* 添加音乐：



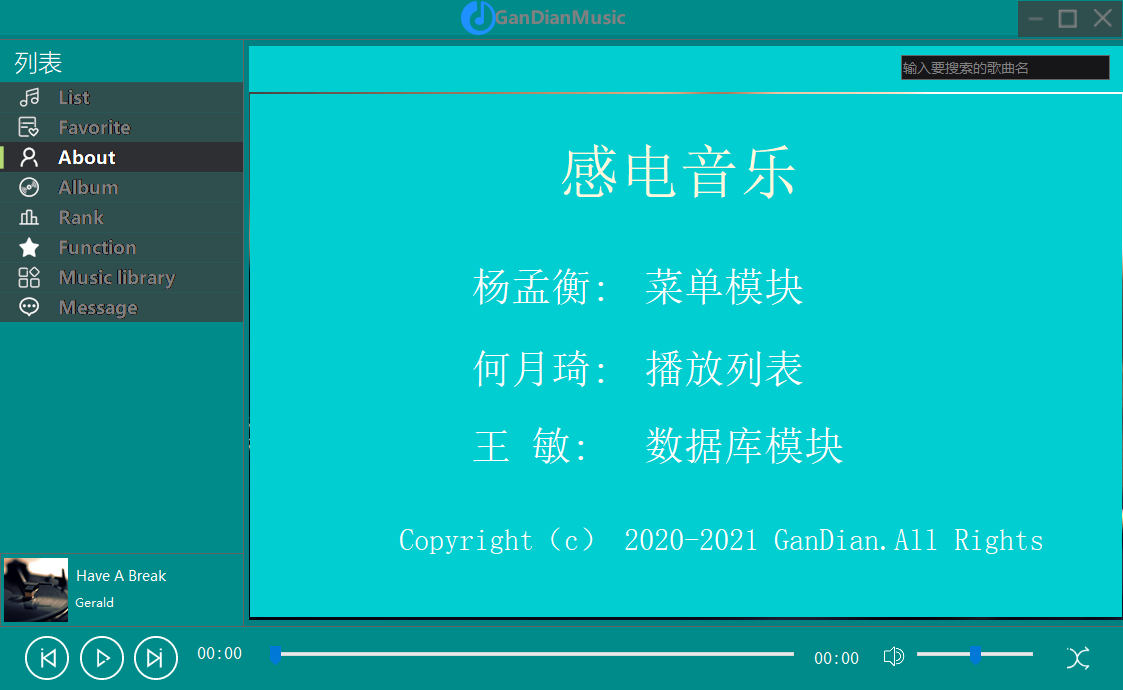




* 删除歌曲：



* 关于页面



* 有很多的实现效果只通过截图来实现不太好展示，故在此链接一个代码演示视频：

<感电音乐演示视频.mp4>

* 之前录制演示视频的时候忘记使用最新版的项目了，因此菜单项about关于页面未曾录制进视频，后来小组成员发现了问题之后我们及时改进，重新录制了一个关于页面的小视频，链接如下：

<感电音乐关于视频.mp4>

# 4.项目小结

通过完成这次C#课程设计、培养了我们的实际分析问题和动手能力，使我们更加充分的理解了课本上所学不到的知识，并能够应用于实践当中。这次课程设计，让我们觉得学术遥远，自己才疏学浅。应用软件的研究何其庞杂，何其精妙，这次设计其实只能是涉其皮毛，距离理想之境还有很长的路。

在制作本项目的过程中，投入了许多的精力与时间。主要是在设计与调节页面上与小组之间配合设计的时间花费的比较多。中间也改了很多次方法逻辑，但是经过小组之间的讨论与思考，最后才一起决定的整体逻辑思路。本项目制作最困难的地方还是在播放列表逻辑切换的地方。之前很多时候会有明明正在播放最爱这个列表。却放到了本地播放列表去。最后经过不断的调试与切换方法才得出能够切换但是不会混乱的效果。除此之外，设计页面ui的过程，以及数据库操作也颇费周折，一边要准备ui素材，一边又要设计ui界面，界面还需要排版，还需要学会对数据库进行操作，小组成员之间也需要学会使用数据库操作，是非常需要交流沟通的问题之一。在紧张的学习生活中，这些项目难点给了我们小组很大的压力，还好最后小组成员们互相交流沟通，互相协助，最终解决了项目难点。

回顾起此次课程设计，从到定题目到顺利完成，从理论到实践，在整个学习的日子里，可以学到很多很多的东西，同时不仅可以巩固了以前所学过的知识，而且还学到了很多在书本上所没有学到过的知识。在设计中遇到了很多难以解决的问题，最后都在大家的辛勤努力下迎刃而解，这种收获是非常难得的，值得珍惜。

本次项目课程设计能够顺利完成，使我们项目小组成员都互相交流学到了许多知识，还有老师在课堂上不倦地指导和教诲，在此深深地感谢他们。感谢辛勤教导我们的老师以及项目小组成员们！