目录

[1需求分析 3](#_Toc486068247)

[1.1功能性需求 3](#_Toc486068248)

[1.2非功能性需求 3](#_Toc486068249)

[1.3约束需求 3](#_Toc486068250)

[2系统设计 4](#_Toc486068251)

[2.1流程图 4](#_Toc486068252)

[2.2框架结构 7](#_Toc486068253)

[模块一：减轻闪烁 7](#_Toc486068254)

[模块二：窗体拖动 8](#_Toc486068255)

[模块三：皮肤选择 8](#_Toc486068256)

[模块四：音量调节 10](#_Toc486068257)

[模块五：歌词显示 12](#_Toc486068258)

[模块六：MP3解析 15](#_Toc486068259)

[模块七：播放与暂停 15](#_Toc486068260)

[模块八：本地和喜爱列表切换 17](#_Toc486068261)

[模块九：拉动条 19](#_Toc486068262)

[模块十：音乐 20](#_Toc486068263)

[2.3数据库 22](#_Toc486068264)

[程序截图 23](#_Toc486068265)

[3总结 24](#_Toc486068266)

[参考文献 24](#_Toc486068267)

# 1需求分析

## 1.1功能性需求

这次作业我做的是一款音乐播放器，播放本地MP3音乐。需要手动添加到程序中，还可以添加歌词。现在的音乐播放器有很多，他们清一色是C/S架构。音乐都在服务器中存放，虽然你也可以下载，但是相信有很多用户，用很多音乐播放器。下载了很多首个，急切的希望有一款本地播放器。来整理电脑上的音乐，整合在一个程序中，在任何时候，即使没有网络连接。也可以放自己想听的音乐，现在WZWMusicPlayer问世啦，你可以对本地音乐进行管理，添加本地音乐，将所有下载的音乐，整合到一个程序中，你可以注册多用户，用户可以设置自己喜爱的播放列表，只需点击小星星即可添加到自己的喜爱列表。列表存放在本地数据库中，随时登录随时听歌！

## 1.2非功能性需求

WZWMusicPlayer拥有堪比商业软件的UI界面。可以跟换皮肤，使用已有的皮肤或者你自己喜欢的图片。所有的图标均来自阿里巴巴图标库，商业一流的图标。配上美丽的背景，让您在听音乐的同时也赏心悦目。（PS：同时感谢阿里巴巴图标库的工作人员，开源了如此多如此美妙的图标）

## 1.3约束需求

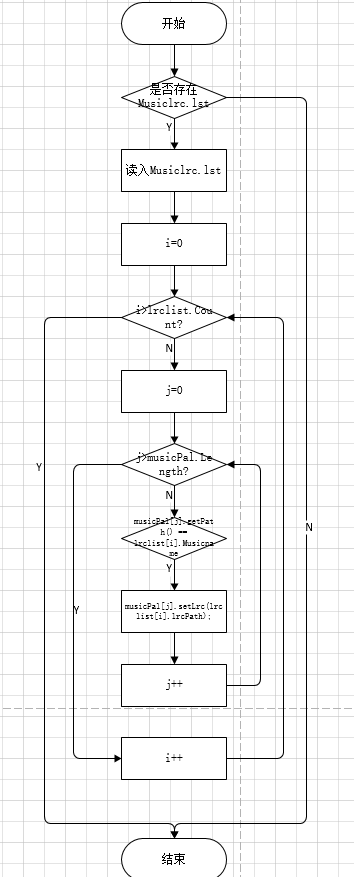
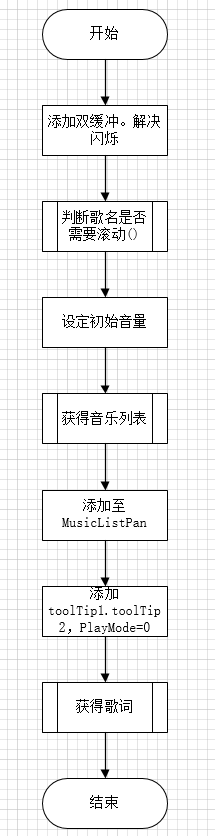
本软件需运行在 .NET Framework 4.5.2的平台上，低平台可能会运行失败，数据库使用的是mysql，需要用户自己建立数据库，并添加相应的表，（在数据库模块会给出sql的导出文件），并更改连接字符串

# 2系统设计

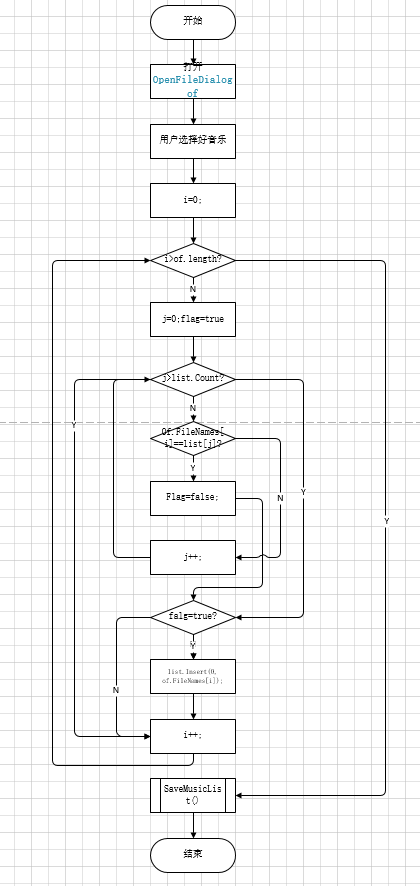
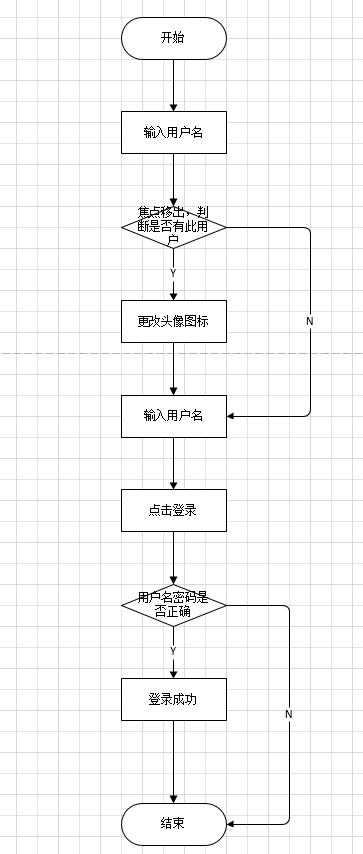
## 2.1流程图

由于系统比较庞大，选了几个比较有代表的流程，画了画。

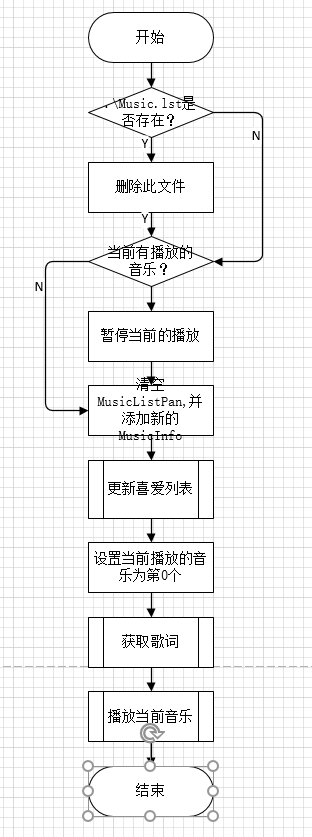
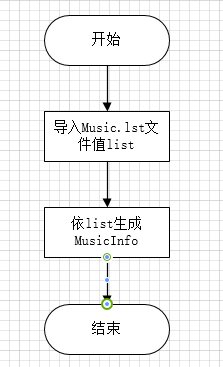
窗体加载 添加歌词



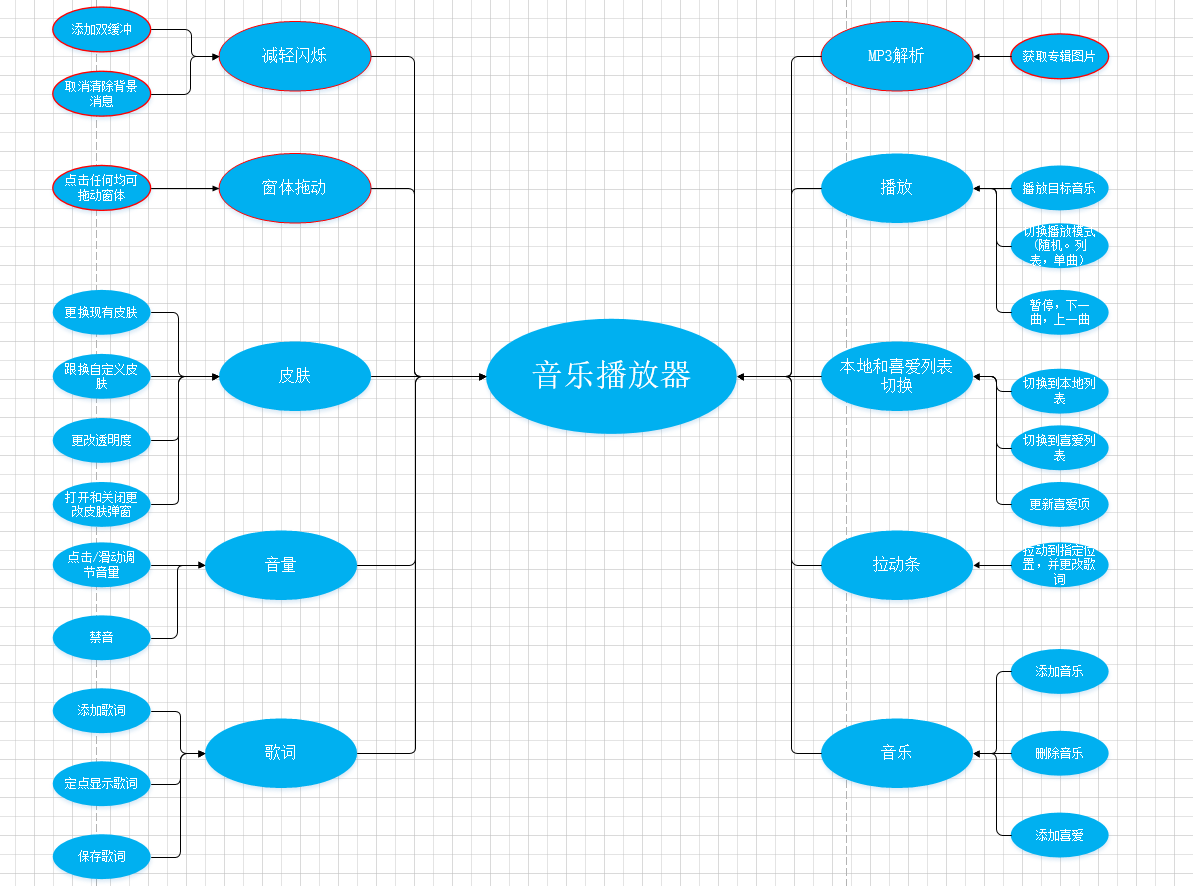
登录 添加音乐



保存音乐 获取音乐

## 2.2框架结构



我的音乐播放器总共分为10个模块:

模块一：减轻闪烁

这个模块包含的小模块有：

1. **添加双缓冲**
2. **取消清除背景**

首先说说双缓冲，双缓冲能有效的降低闪烁，计算机图形学中，学到过双缓冲机制，就是将下一帧的图像预先存到内存中，当运行至下一帧时，直接从内存中调出显示，这样，节省了再次生成图像所计算的时间，需要的内存空间会变大，计算机的性能也高。程序中，可以直接在Form的属性中设置Form双缓冲，但这不是我想要的，我需要将所有的子控件都设置为双缓冲。这段代码非常简洁：

protected override CreateParams CreateParams

{

get

{

CreateParams cp = base.CreateParams;

cp.ExStyle |= 0x02000000;////用双缓冲从下到上绘制窗口的所有子孙

return cp;

}

}

然后再说说这个取消清除背景，Form在发生改变时会重绘真个界面，同时也会清除背景，这样会带来白色的闪烁，取消掉背景清除也会减轻闪烁。这段代码也非常简洁：

protected override void WndProc(ref Message m)

{

if (m.Msg == 0x0014) // 禁掉清除背景消息

return;

base.WndProc(ref m);

}

模块二：窗体拖动

这个模块包含的小模块有：

1）**点击任何均可拖动窗体**

这里需要引入Win32的api

public const int WM\_NCLBUTTONDOWN = 0xA1;

public const int HTCAPTION = 0x2;

[DllImportAttribute("user32.dll")]

public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

[DllImportAttribute("user32.dll")]

public static extern bool ReleaseCapture();

然后在你想让他拖动的面板或窗体中，添加MouseDown事件，事件执行的代码如下：

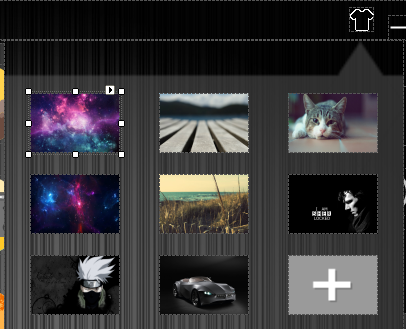
ReleaseCapture();

SendMessage(Handle, WM\_NCLBUTTONDOWN, HTCAPTION, 0);

模块三：皮肤选择

这个模块包含的小模块有：

1. **更换现有皮肤**
2. **跟换自定义皮肤**
3. **更改透明度**
4. **打开和关闭更改皮肤弹窗**

首先一下更**换现有皮肤**，它的实现很简单，首先需要将使用到的图片背景添加到资源中，然后在主窗体中添加一个弹出框，其中再添加几个PictureBox，再为PictureBox添加点击事件，事件内容是将主窗体的背景图片改为相应的图片。就像这样：

点击PictureBox事件代码为：

this.BackgroundImage = Properties.Resources.背景;

然后我们在说说**更换自定义的皮肤**，你看到上面的图片右下角有个加号的的图标吗。其实也是一个PictureBox，为其添加点击事件，如下：

private void Skin\_add\_Click(object sender, EventArgs e)

{

OpenFileDialog of1 = new OpenFileDialog(); //打开文件窗口

of1.InitialDirectory = "C:\\Users\\Lenovo\\Pictures\\Camera Roll\\"; //开始文件位置

of1.Filter = "图片|\*.jpg;\*.png;\*.gif;\*.jpeg;\*.bmp"; //文件过滤 名称：图片

of1.RestoreDirectory = true; //关闭前还原原目录

of1.FilterIndex = 1; //选定文件索引器下标

if (of1.ShowDialog() == DialogResult.OK)

{

picfile = of1.FileName;

this.BackgroundImage = Image.FromFile(string.Format("{0}", picfile));

}

}

OpenFileDialog是我们经常使用打开文件的系统弹窗，可以使用ofl.Filter过滤文件，只显示图片格式的文件如\*.jpg,\*.png等，DialogResult.OK即为我们点击了确定按钮，或双击了选中的文件。这时获取文件路径，将主窗体背景背景更改为选中的图片。

接下来我们要说**更改透明度**了，这个的实现我是用了，一个panel，一个复写的ProgressBar,一个PictureBox，为三个组件都添加。MouseDown，MouseMove，MouseUp，事件，他的样子是这样的：

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\透明度滑块.bmp

前两个组件的事件都一样，首先鼠标控制事件，我认为分两种，第一种是点击任意位置一下，改变为对应位置的透明度值，第二种是鼠标拖动滑块移动到任意位置，同时改变透明度。

所以，当MouseDown时，就要获取鼠标的位置，因为鼠标在panel里，所以鼠标的位置是相对一panel的，我们可以获得鼠标的X坐标。它的值即为我们ProgressBar的Value,并更改form的透明度属性。同时设置鼠标按下的Flag为true。代码如下:

public void setTrans(int op)

{

this.Opacity = (double)op / 220;

}

bool Transflag = false;

private void TransPan\_MouseDown(object sender, MouseEventArgs e)

{

TransPro.Value = e.Location.X;

setTrans(e.Location.X);

int x2 = e.Location.X - 6;

if (x2 < 0) { x2 = 0; }

if (x2 > 207) { x2 = 207; }

TransPoint.Location = new Point(x2, TransPoint.Location.Y);

Transflag = true;

}

接下来就该鼠标移动了，我们上面设置的Flag就是为移动做准备的，只有当鼠标按下时才可以滑动滑块。改变透明度方法和上面一样，不过因为我们有个PictureBox当滑块，它也有大小，它的中心位置在ProgressBar的Value点上，所有需要对它进行一些处理，不要让它消失在panel中。鼠标的X坐标也肯是负数同样要做判断。

private void TransPro\_MouseMove(object sender, MouseEventArgs e)

{

int x = 0, x2 = 0;

if (Transflag)

{

x = e.Location.X;

x2 = e.Location.X - 6;

if (x < 0) { x = 0; }

if (x > (220)) { x = 220; }

if (x2 < 0) { x2 = 0; }

if (x2 > 207) { x2 = 207; }

TransPro.Value = x;

setTrans(x);

TransPoint.Location = new Point(x2, TransPoint.Location.Y);

}

}

当鼠标抬起时需要将鼠标按下的flag置为false;

对于PicturBox的事件，因为鼠标是在PictureBox中，所以鼠标的位置是相对于这个PictureBox的，我们不能直接用它的值，需要进行一些转换，首先获得全局鼠标位置，然后在通过函数得到相对于Panel的位置。

//鼠标相对于屏幕的坐标

p3 = MousePosition;

//鼠标相对于窗体的坐标

p4 = TransPan.PointToClient(p3);

**打开和关闭更改皮肤弹窗**其实就是，设置点击皮肤小图标显示 这个弹窗，再次点击隐藏这个弹窗，使用一个bool flag控制其有无。

模块四：音量调节

这个模块包含的小模块有：

1. **点击/滑动调节音量**
2. **禁音**

首先说**点击/滑动调节音量**，暂时没有找到如何让ProgressBar旋转的方法，所以我使用了，两个panel，第一个panel背景色为黑色，第二个panel背景色为白色，第二个覆盖第一个。为这两个panel添加MouseDown，MouseMove，MouseUp事件，实现与上面

的透明度滑块相同，但是第二个panel的点击后得到的Y值，panel2.Size.Height-Y 才是他新的大小，而他的位置，则为panel1.Size.height-新的大小。他的样子是这样的：



更改音量大小的代码如下：

private void VolumUpPan\_MouseDown(object sender, MouseEventArgs e){

int rx = VolumUpPan.Size.Height;

VolumUpPan.Size = new Size(5, rx - e.Location.Y);

VolumUpPan.Location = new Point(0, 150 - (rx - e.Location.Y));

setVolume(rx - e.Location.Y, 150);

Volumflag = true;}

private void setVolume(int Y, int A){

double p = (double)Y / A;

per = (int)(p \* 100);

if (isSilence){this.axWindowsMediaPlayer1.settings.volume = 0;}

else{this.axWindowsMediaPlayer1.settings.volume = per;}

this.VolumePersentLab.Text = per.ToString() + "%";

}

需要补充一点，因为系统默认是以左上角为原点，所以，对于第一个panel的点击后得到的Y值，Panel1.size.Height-Y 才是第二个panel的大小：

private void VolumeBackPan\_MouseDown(object sender, MouseEventArgs e)

{

VolumUpPan.Size = new Size(5, 150 - e.Location.Y);

VolumUpPan.Location = new Point(0, e.Location.Y);

setVolume(150 - e.Location.Y, 150); Volumflag = true;

}

**禁音：**禁音是点击滑块下面的音量小图标，将音量设置为0,并更改图片;

private void SilenceBut\_Click(object sender, EventArgs e){

if (isSilence){

SilenceBut.BackgroundImage = Properties.Resources.声音\_\_1\_;

volumeBut.BackgroundImage = Properties.Resources.声音\_\_1\_;

SilenceSetVolume();

isSilence = false;

}

else{

SilenceBut.BackgroundImage = Properties.Resources.静音\_\_1\_;

volumeBut.BackgroundImage = Properties.Resources.静音\_\_1\_;

isSilence = true;

this.axWindowsMediaPlayer1.settings.volume = 0;

}

}

模块五：歌词显示

这个模块包含的小模块有：

1. **添加歌词**
2. **定点显示歌词**
3. **保存歌词**

**添加歌词**，打开一个文件选择框，选中歌词lrc文件。因为我的添加歌词按钮是在，我另写的一个用户控件，所以需要将主窗体指针传入用户控件，在用户控件中调用主窗体的public方法，保存歌词。

private void PlusGeCiBut\_Click(object sender, EventArgs e) //添加歌词

{

OpenFileDialog of = new OpenFileDialog(); //打开一个文件窗口

of.InitialDirectory = "C:\\Users\\Lenovo\\Music"; //默认打开位置

of.Filter = "歌词|\*.lrc"; //文件过滤器

of.RestoreDirectory = true;

of.FilterIndex = 1;

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

{

lrcPath = of.FileName;

}

this.fatherForm.SaveMusicLrcList(this.lrcPath, this.MusicPath);

if (fatherForm.getNowPlaying() == this||fatherForm.getNowPlaying()==null)

{

fatherForm.setGeCi(lrcPath);

}

}

**定点显示歌词**，首先先看一看lrc歌词文件的格式：

[ti:红玫瑰]

[ar:陈奕迅]

[al:THE FIRST ELEVEN YEARS然后呢 1997-2007跨世纪国语精选 / The 1st Eleven ]

[by:]

[offset:0]

[00:01.00]红玫瑰 (《白玫瑰》国语版) - 陈奕迅

[00:03.93]词：李焯雄

[00:06.43]曲：梁翘柏

[00:08.61]编曲：梁翘柏

[00:11.05]

[00:16.28]梦里梦到醒不来的梦

[00:18.65]红线里被软禁的红

[00:23.53]所有刺激剩下疲乏的痛

可以看到，它分为两部分，[]中一部分，后面一部分，[]中是在什么时间显示它，后面是歌词。所以这就好办了，只需写一个正则表达式去匹配获得这两部分即可，需要注意的一点是，时间00:23.53 最后一位是. 我把他替换成了: 这样有利于split 它，从而计算double值的时间，还要一点是，开头部分是一些无关歌词的部分，需要在匹配时去掉他们。

正则匹配获得这两部分的数组的代码如下：

match字符串：

string excTime = @"(?<=\[).\*?(?=\])";//匹配时间的正则

string excText = @"(?<=\])(?!\[).\*";//匹配歌词的正则

public void getLrc(string FileName){

try{

t1 = 0;

t2 = 0;

string[] strs = System.IO.File.ReadAllLines(FileName, Encoding.GetEncoding("GB2312"));

lrcTime = new string[strs.Length];

lrcText = new string[strs.Length];

int hasline = strs.Length;

MatchCollection match1;

MatchCollection match2;

for (int i = 0; i < hasline; i++){

if (strs[i].StartsWith("[ti:") | strs[i].StartsWith("[ar:") | strs[i].StartsWith("[al:") | strs[i].StartsWith("[by:") | strs[i].StartsWith("[offset:")) { }

else{

match1 = Regex.Matches(strs[i], excTime);

match2 = Regex.Matches(strs[i], excText);

foreach (var v in match1){

StringBuilder sb = new StringBuilder(v.ToString());

sb.Replace(".", ":");

zj = sb.ToString();

try{

lrcTime[t1] = zj;

foreach (var t in match2){

lrcText[t2] = t.ToString();

}

t1++;

t2++;

} catch (Exception e1){Console.WriteLine(e1.Message);}

}

}

}

}catch (Exception e){Console.WriteLine(e.Message);}

}

**保存歌词**：就是将歌词路径和对应的音乐路径写入文件，这里我使用了一个可序列化的类：

[Serializable()]

class MusicandLrc

{

public String Musicname;

public String lrcPath;

}

先找查找，对应的音乐，是否有歌词，如果有，则删除原来的，添加新的，最后写入文件。

public void SaveMusicLrcList(String lrcpath, String musicname) //保存至.lst文件

{

MusicandLrc m = new MusicandLrc();

m.lrcPath = lrcpath;

m.Musicname = musicname;

int num = -1;

for (int i = 0; i < lrclist.Count; i++)

{

if (musicname.Equals(lrclist[i].Musicname))

{

num = i;

}

}

if (num != -1)

{

lrclist.RemoveAt(num);

}

lrclist.Add(m);

if (File.Exists(".\\Musiclrc.lst") == true)

{

File.Delete(".\\Musiclrc.lst");

}

SaveFileDialog sf = new SaveFileDialog();

sf.FileName = "Musiclrc.lst";

sf.RestoreDirectory = true;

sf.FilterIndex = 1;

FileStream fs = new FileStream(string.Format("{0}", sf.FileName), FileMode.Create);

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(fs, lrclist); //保存

fs.Close();

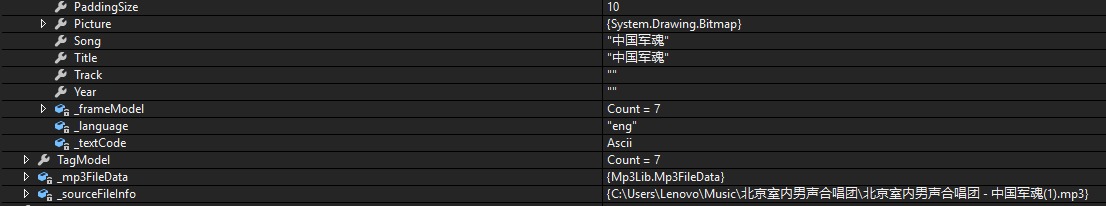
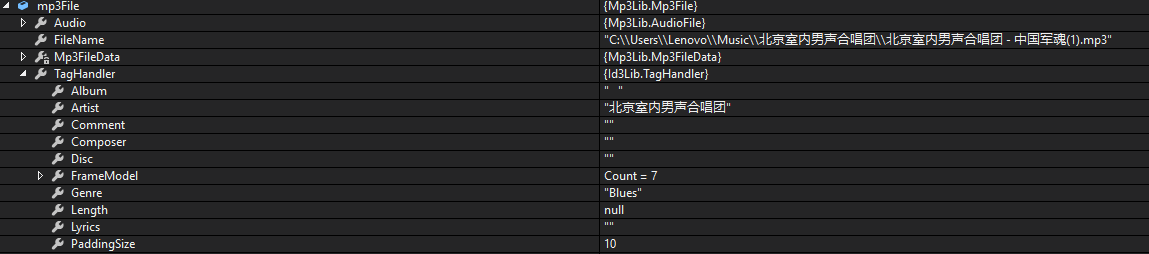
}

模块六：MP3解析

这个模块包含的小模块有：

1. **获取专辑图片**

**获取专辑图片**：调用了一个解析MP3文件的项目，MP3文件有一个id3标签，这个标签中存放着，这个MP3的一些信息，如歌手，专辑名，音乐类型，专辑图片等等。调用这个项目的Mp3File.TagHandler成员可以得到这些信息：



代码如下：

public void ChangePicture(string filename){

Mp3File mp3File = null;

try{

mp3File = new Mp3File(filename);

//新建Mp3File

this.MusicPpicture.BackgroundImage = mp3File.TagHandler.Picture;

//获得专辑图片

}catch (Exception e){Console.WriteLine(e.Message);}

}

模块七：播放与暂停

这个模块包含的小模块有：

1. **播放目标音乐**
2. **切换播放模式（随机，列表，单曲）**
3. **暂停，下一曲，上一曲**

播放目标音乐：我的播放是使用AxWindowsMediaPlayer播放

public void play(string namepath) //播放

{

try

{

GeCi.Text = "";

this.axWindowsMediaPlayer1.URL = namepath; //设置播放路径

this.axWindowsMediaPlayer1.Ctlcontrols.currentPosition = 0;

haveplayed = true; //将已有播放设为true

MusicName.Text = this.axWindowsMediaPlayer1.currentMedia.name; //设置歌曲名称

MusicName2.Text = this.axWindowsMediaPlayer1.currentMedia.name;

ChargeRoll(); //判断歌名是否需要滚动显示

isplay = true; //设置正在播放为true

timer2.Enabled = true; //播放时间timer 开启

TotalMusicTimeLab.Text = this.axWindowsMediaPlayer1.currentMedia.durationString; //设置播放总时长

PlayBut.BackgroundImage = Properties.Resources.播放\_\_8\_; //改变播放按钮

for (int i = 0; i < musicPal.Length; i++)

{

if (musicPal[i].getPath() == namepath)

{

nowPlayIndex = i;

break;

}

}

if (musicPal[nowPlayIndex].getLrc() != null)

{

setGeCi(musicPal[nowPlayIndex].getLrc());

}

else

{

stopGeCi();

}

ChangePicture(namepath);

}catch (Exception e){Console.WriteLine(e.Message);}

}

**切换播放模式**：就是在axWindowsMediaPlayer1播放完毕的时候。下一首播放哪一个音乐。一共有四种模式，第一种是列表循环，第二种是单曲循环，第三种是随机播放，第四种是播放一次。使用一个int型变量控制播放模式：

if (this.axWindowsMediaPlayer1.playState.ToString() == "wmppsStopped")

{

try

{

switch (PlayMode)

{

case 0: this.LoopPlay(); break;

case 1: this.SinglePlay(); break;

case 2: this.RandomPlay(); break;

case 3: this.stop(); break;

default: this.stop(); break;

}//播放下一首

}catch (Exception) {timer2.Enabled = false; }

}

**暂停，下一曲，上一曲**：就是音乐播放器都有的功能：



需要注意的是不光要更改主窗体的图标，同时还要更改音乐列表的图标。

模块八：本地和喜爱列表切换

这个模块包含的小模块有：

1. **切换到本地列表**
2. **切换到喜爱列表**
3. **更新喜爱项**

**切换到本地列表**：基本思想是，清空当前音乐列表，添加本地音乐列表，如果有正在播放的音乐，更改其图标及flag。更新喜爱。

public void changelisttobd(){

if (!leftchooseflag) {

this.PlusMusicBut.Visible = true;

this.MusicListPan.Controls.Clear(); //清空

if (nowPlaying != null) {

nowPlayingmusic = nowPlaying.getPath();

}

this.musicPal = new MusicInfo[this.list.Count];

int locx = 0;

int locy = 0;

for (int i = 0; i < list.Count; i++){

musicPal[i] = new MusicInfo();

Point p = new Point(locx, locy);

locy += 50;

musicPal[i].Location = p;

musicPal[i].setFatherForm(this);

musicPal[i].setMusicNameAndPath(getFileName(list[i]), list[i]);

this.MusicListPan.Controls.Add(musicPal[i]);

}

if (nowPlayingmusic != null) //检查是否正在播放音乐 若是更改相应的MusicInfo{

int num = -1;

for (int i = 0; i < musicPal.Length; i++){

if (nowPlayingmusic == musicPal[i].getPath()){

num = i;

}

}

if (num != -1) {

this.setNowPlaying(musicPal[num]);

this.nowPlayIndex = num;

GetLrc();

this.nowPlaying.changePlayBut(true);

}else{

this.nowPlayIndex = -1;

}

}

UpdateLike();

leftchooseflag = true;

panel4.BackColor = Color.Gray;

panel5.BackColor = Color.Transparent;

}

}

**切换到喜爱列表**：基本思想和切换到本地列表相同，只是添加的列表为喜爱列表，只有登录后才可点击。

public void changelisttolike(){

if (!loginflag) {

DialogResult dr = MessageBoxEx.Show(this, "未登录");

}else{

if (leftchooseflag) {

this.PlusMusicBut.Visible = false;

this.MusicListPan.Controls.Clear();

if (nowPlaying != null) {

nowPlayingmusic = nowPlaying.getPath();

}

this.musicPal = new MusicInfo[this.likeMusicPath.Count];

int locx = 0;

int locy = 0;

for (int i = 0; i < likeMusicPath.Count; i++){

musicPal[i] = new MusicInfo();

Point p = new Point(locx, locy);

locy += 50;

musicPal[i].Location = p;

musicPal[i].setFatherForm(this);

musicPal[i].setMusicNameAndPath(getFileName(likeMusicPath[i]), likeMusicPath[i]);

musicPal[i].DeleteBut.Visible = false;

his.MusicListPan.Controls.Add(musicPal[i]);

}

if (nowPlayingmusic != null) {

int num = -1;

for (int i = 0; i < musicPal.Length; i++){

if (nowPlayingmusic == musicPal[i].getPath()){

num = i;

}

}

if (num != -1) {

this.setNowPlaying(musicPal[num]);

this.nowPlayIndex = num;

GetLrc();

this.nowPlaying.changePlayBut(true);

}else {

this.nowPlayIndex = -1;

}

}

UpdateLike();

leftchooseflag = false;

panel5.BackColor = Color.Gray;

panel4.BackColor = Color.Transparent;

}

}

}

**更新喜爱项**：就是对列表进行遍历，如果Islike==true则改变其图标。

模块九：拉动条

这个模块包含的小模块有：

1. **拉动到指定位置，并更改歌词**

拉动条的实现和透明度的基本相同。只是对得到的数据进行不同的操作，这里是，将数据换算成，对应的double时间位置:

private double getDoubleTime(string time) //用时间字符串解析为double时间 01:32:34 分：秒：毫秒 34\*0.01+32+1\*60

{

string[] t = time.Split(':');

if (t[0].First() == '0') {

t[0].Remove(0, 1);

}

int m = int.Parse(t[0]);

if (t[1].First() == '0') {

t[1].Remove(0, 1);

}

int s = int.Parse(t[1]);

if (t[2].First() == '0') {

t[2].Remove(0, 1);

}

int w = int.Parse(t[2]);

double sum = (double)m \* 60 + (double)s + (double)w / 100;

return sum;

}

用axwindowsMediaPlayer获取总共播放时间：

double b = x / all;

double alltime = this.axWindowsMediaPlayer1.currentMedia.duration;

double thisTime = alltime \* b;

更改歌词就是，去搜索得到的解析后的歌词数组，找到第一个时间大于当前时间的位置，它的前一个位置的歌词即为当前要显示的歌词。当为0时不显示：

for (int i = 0; i < lrcTime.Length; i++){

if (lrcTime[i] != null) {

if (getDoubleTime(lrcTime[i]) > this.axWindowsMediaPlayer1.Ctlcontrols.currentPosition) {

if (i != 0) {

if (GeCi.Text != lrcText[i - 1]) {

GeCi.Text = lrcText[i - 1];

ChargeRoll2();

}

}

break;

}

}

}

模块十：音乐

这个模块包含的小模块有：

1. **添加音乐**
2. **删除音乐**
3. **添加喜爱**

**添加音乐**：打开文件选择窗口，选择音乐后，先和已有的列表进行比对，忽略掉重复的音乐，添加没有的音乐，最后在保存至文件，更新音乐列表，播放添加的音乐。

点击添加按钮事件：

private void PlusMusicBut\_Click(object sender, EventArgs e) //添加音乐

{

OpenFileDialog of = new OpenFileDialog(); //打开一个文件窗口

of.InitialDirectory = "C:\\Users\\Lenovo\\Music"; //默认打开位置

of.Filter = "音乐|\*.mp3;\*.wav;\*.wma"; //文件过滤器

of.RestoreDirectory = true;

of.FilterIndex = 1;

of.Multiselect = true; //多选择

if (of.ShowDialog() == DialogResult.OK) {

for (int i = 0; i < of.FileNames.Length; i++){

bool flag = true;

for (int j = 0; j < list.Count; j++) {

if (of.FileNames[i] == list[j]) {

flag = false;

break;

}

}

if (flag) {

list.Insert(0, of.FileNames[i]);

}

}

SaveMusicList();

}

}

添加音乐至文件：

public void SaveMusicList() //保存至.lst文件

{

if (File.Exists(".\\Music.lst") == true) {

File.Delete(".\\Music.lst");

}

SaveFileDialog sf = new SaveFileDialog();

sf.FileName = "Music.lst";

sf.RestoreDirectory = true;

sf.FilterIndex = 1;

FileStream fs = new FileStream(string.Format("{0}", sf.FileName), FileMode.Create);

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(fs, list); //保存

fs.Close();

if (this.nowPlaying != null) //跟新播放列表

{

this.nowPlaying.pause();

}

this.MusicListPan.Controls.Clear();

this.musicPal = new MusicInfo[list.Count];

int locx = 0;

int locy = 0;

for (int i = 0; i < list.Count; i++){

musicPal[i] = new MusicInfo();

Point p = new Point(locx, locy);

locy += 50;

musicPal[i].Location = p;

musicPal[i].setFatherForm(this);

musicPal[i].setMusicNameAndPath(getFileName(list[i]), list[i]);

this.MusicListPan.Controls.Add(musicPal[i]);

}

UpdateLike();

this.setNowPlaying(musicPal[0]);

this.nowPlayIndex = 0;

GetLrc();

this.nowPlaying.play();

this.play(musicPal[0].getPath());

}

**删除音乐**：需要判断删除的是否为正在播放的，如果是，设置正在播放为第0个音乐，如果不是继续播放，清空列表，添加删除后的列表，设置属性，如果被删光了音乐，暂停播放，将专辑图片改为默认。

**添加喜爱**：只有登录后才可以添加喜爱，点击后添加到喜爱列表，在退出或关闭未退出而音乐播放器时，写入数据库，值得注意的是，转义字符，因为存的是路径所以有”\\”，需要将路径先以”\\”分割在将对应的”\\”，换成”\\\\”；再存入数据库。这样数据库里的字符串就是C#可用的路径。

额外模块

**登录**：登录模块是为了使用数据库而加入的模块，用户可以创建账号，选择头像，登录，登陆后可以添加喜爱的音乐。

## 2.3数据库

user表：

|  |  |  |
| --- | --- | --- |
| user | 用户表 |  |
| 列名 | 属性 | 描述 |
| user\_name | varchar(50) | 用户名 主键 |
| password | varchar(50) | 密码 |
| image\_num | int | 头像号（暂定1-9） |

Sql语句：

DROP TABLE IF EXISTS `user`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `user` (

`user\_name` varchar(50) NOT NULL,

`password` varchar(50) NOT NULL,

`image\_num` int(11) NOT NULL,

PRIMARY KEY (`user\_name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

user\_like\_music\_list表：

|  |  |  |
| --- | --- | --- |
| user | 用户表 |  |
| 列名 | 属性 | 描述 |
| like\_list\_id | int | 自增 主键 |
| user\_name | varchar(50) | 用户名 主外键 |
| music\_path | varchar(400） | 音乐路径 |

DROP TABLE IF EXISTS `user\_like\_music\_list`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `user\_like\_music\_list` (

`like\_list\_id` int(11) NOT NULL AUTO\_INCREMENT,

`user\_name` varchar(50) NOT NULL,

`music\_path` varchar(400) NOT NULL,

PRIMARY KEY (`like\_list\_id`),

KEY `user\_name` (`user\_name`),

CONSTRAINT `user\_like\_music\_list\_ibfk\_1` FOREIGN KEY (`user\_name`) REFERENCES `user` (`user\_name`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=116 DEFAULT CHARSET=utf8;

# 程序截图









# 3总结

整个程序写了两个星期，遇到不会的问题我会去Google或者查书，比如正则表达，减轻闪烁，MP3解析，学到了很多。看着小程序一点一点变得壮大。很有成就感。总之，作为大学课程里最后的一门课设，我真的下了很大的功夫，之后希望把它更加完善，比如扩展成C/S架构，可以从服务器中查找音乐等等。

# 参考文献

C#高级编程（第八版） [美] Christian Nagel，Jay Glynn，Morgan Skinner 著；李铭 译

head first C#（第三版） [美] 安德鲁·施特尔曼（Andrew Stellman） 著

C#入门经典（第五版） [美] Karli Watson，Jacob Vibe Hammer，Jon D Reid 等 著；齐立波，黄俊伟 译