MusicPlayer项目说明文档

> 同济大学2019级软件学院数据结构课荣誉班项目

项目信息

- **项目名称：用Qt实现一个轻量级的音乐播放器**
- **指导老师：张颖老师**
- **小组成员：1951576　沈星宇**
  1953288　张思源
- **贡献比例：沈星宇：100%**
  张思源：100%
- **项目github地址：**

  https://github.com/Sherlock-White/2020-TJ-program-MusicPlayer
- **参考博客：**

  https://blog.csdn.net/Kingsman_T/article/details/103879947

项目开发文档

- 开发环境：Qt5 （Qt Creator 4.13.3 + Qt 5.9.9 MinGW 32bit）
- 支持平台：windows
- 完成时间：2020.12.6 - 2020.12.31

1 功能分析

基础功能：

- [x] 支持添加本地文件（支持格式：.mp3 、.flac 、.mpga ）并播放
- [x] 支持解析歌词以滚动形式展示
- [x] 支持解析专辑封面
- [x] 支持自定义主页面背景
- [x] 支持歌单管理（”我喜欢“+”其他自定义歌单“）
- [x] 支持多种歌曲循环模式
- [x] 支持“下一曲、下一曲”
- [x] 支持“暂停、播放”

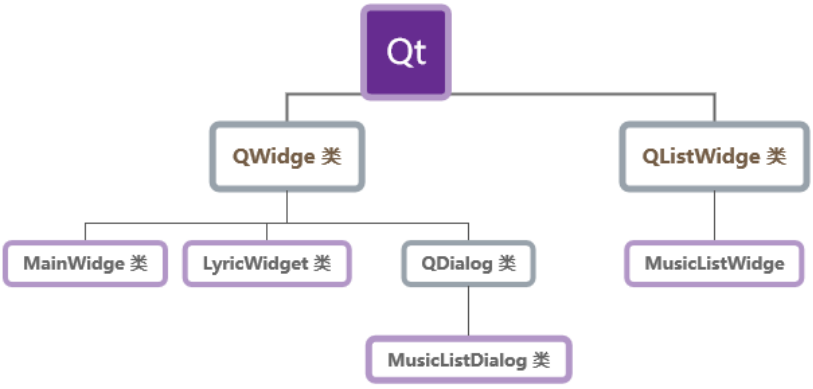
细节功能：

- [x] 鼠标悬浮在按钮上时有提示信息
- [x] 任务栏右击进行任务选择
- [x] 主页面通过”about“来展示项目基本信息

2 类结构设计

项目中用到的类如下：

- Music 类
- MusicList 类
- MusicListWidget 类
- MusicListDialog 类
- LyricWidget 类
- MainWidget 类

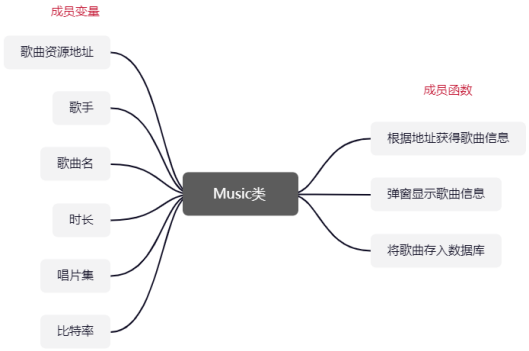


解释：灰色边框的类是Qt提供的可继承的类，紫色边框的为本项目从原有类中派生出来的类，详细的接口、成员与功能在2.2阐述。

2.2 各个类详细展示

2.2.1 Music 类（歌曲类）

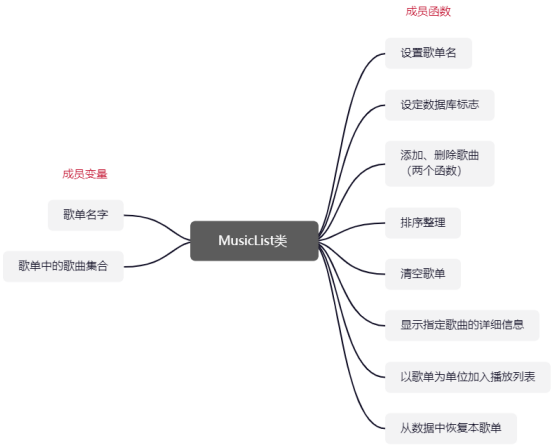
> 用来储存歌曲的信息，包含数据库的存取和歌曲解析两大功能




```
class Music
{
private:
    QUrl url;                   //歌曲资源地址
    QString author;             //歌手
    QString title;              //歌曲名
    qint64 duration;            //时长
    QString albumTitle;         //唱片集
    int audioBitRate;           //比特率
    void refreshInfo();         //根据歌曲url获得歌曲信息
    friend class MusicCompare;  //用于歌曲排序的函数
public:
    Music(){}
    Music(QUrl iurl);
    QUrl getUrl() const {return url;
    QString getInfo() const;    //返回歌曲的信息
    void detail();              //弹窗显示歌曲信息
    void insertSQL(const QString& name);
                                //存入数据库
    QString getLyricFile();     //根据文件名来获取歌词路径
    friend class MusicList;
};
```

2.2.2 MusicList 类（歌单类）

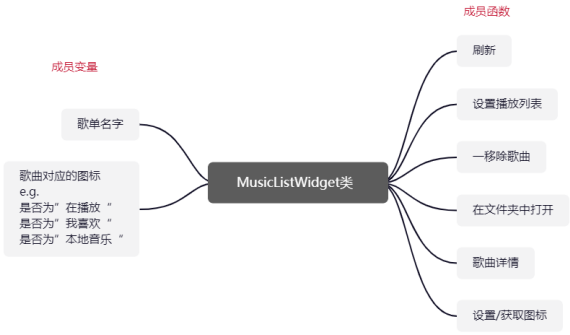
> 歌曲列表（即歌单），是用容器vector将music类的对象存储起来，由此可以进行一些具体的、针对多首歌曲的操作




```cpp
class MusicList
{

    QString name;            //歌单名

    vector<Music> music;    //所存储的歌曲


    bool sql_flag=true;     //控制是否需要与数据库交互
                            //e.g.删除播放列表的歌曲时不需要更新数据库
    friend class MainWidget;//设置友元关系，便于加入主页面中
public:
    MusicList(){}
    MusicList(const QList<QUrl>& urls,QString iname="");

    void setName(const QString& iname){name=iname;}
                                        //设定歌单名
    QString getName(){ return name; }   //设定数据库标志
    void setSQL(bool on){ sql_flag=on; }    //从url添加歌曲
    void addMusic(const QList<QUrl>& urls);
    void addMusic(const Music& iMusic);     //添加一首歌曲
    Music getMusic(int pos);                //获取指定位置的歌曲
    void addToPlayList(QMediaPlaylist *playlist);
                                        //将本歌单加入播放列表
    void addToListWidget(MusicListWidget *listWidget);
                                        //歌单可视化
    void removeMusic(int pos);          //移除指定的歌曲
    void showInExplorer(int pos);       //在文件夹中打开
    void detail(int pos);               //显示指定歌曲详细信息
    void remove_SQL_all();              //数据库中移除全部本歌单的歌曲
    void insert_SQL_all();              //将歌单中的歌曲全部写入数据库
    void read_fromSQL();                //从数据中恢复本歌单
    void sort_by(COMPARE key);          //将本列表中的歌曲排序
    void neaten();                      //整理歌单：将歌单中的重复歌曲去掉并排序
    void clear();                       //清空本歌单
};
```

2.2.3 MusicListWidget 类（歌单窗口）

> 用来展示歌单（其实还包括”播放列表“和”本地音乐“），主要是将歌单能实现的功能与窗口上的按钮建立信号槽




```cpp
class MusicListWidget:public QListWidget
{
    Q_OBJECT
    MusicList musicList;             //当前歌曲列表（存储的是歌曲信息）
    QIcon icon=QIcon(":/image/image/image/music.png");
                                     //当前展示列表项使用的图标
public:
    MusicListWidget(QWidget *parent = Q_NULLPTR);
    void refresh();                  //刷新显示（当musicList有所变化时，需要调用）
    void setMusicList(const MusicList& music);
                                     //设置歌曲列表
    void setMusicList_playing(const MusicList& music);
                                     //设置播放列表
    void removeMusic();              //移除歌曲
    void showInExplorer();           //在文件夹中打开
    void detail();                   //歌曲详情
    void setIcon(QIcon iicon){ icon=iicon; }
                                     //设置/获取图标Icon
    QIcon getIcon(){ return icon; }
    friend class MainWidget;         //建立友元关系，便于在主页面中插入
};
```

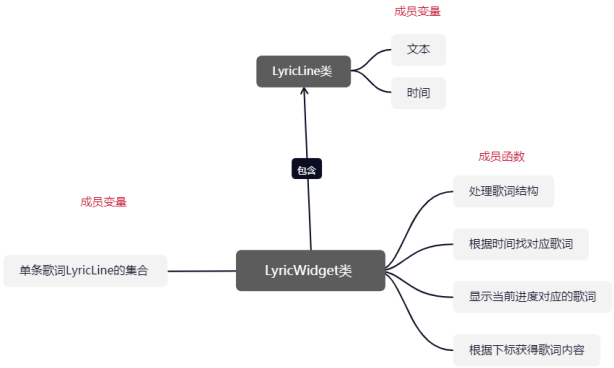
2.2.4 MusicListDialog 类（”添加至歌单“时弹出的对话框）

> 歌单的创立需要用户自行选择歌曲加入歌单，此时将弹出一个窗口，供用户从”本地音乐“中选取一个或多个歌曲进入对应的歌单


```cpp
class MusicListWidget:public QListWidget
{
    Q_OBJECT
    MusicList musicList;             //当前歌曲列表（存储的是歌曲信息）
    QIcon icon=QIcon(":/image/image/image/music.png");
                                     //当前展示列表项使用的图标
public:
    MusicListWidget(QWidget *parent = Q_NULLPTR);
    void refresh();                  //刷新显示（当musicList有所变化时，需要调用）
    void setMusicList(const MusicList& music);
                                     //设置歌曲列表
    void setMusicList_playing(const MusicList& music);
                                     //设置播放列表
    void removeMusic();              //移除歌曲
    void showInExplorer();           //在文件夹中打开
    void detail();                   //歌曲详情
    void setIcon(QIcon iicon){ icon=iicon; }
                                     //设置/获取图标Icon
    QIcon getIcon(){ return icon; }
    friend class MainWidget;         //建立友元关系，便于在主页面中插入
};
```

2.2.5 LyricWidget 类（歌词显示的窗口）

> 用来专门显示歌词的窗口，嵌入至主页面中，点击”下一曲“旁边标注有”词“的按钮即可召唤此窗口，建立在”单条歌词（LyricLine类）“的基础上




```cpp
class LyricLine
{
public:
    qint64 time;
    QString text;
    LyricLine(qint64 time, QString text):time(time), text(text){}
};

class LyricWidget : public QWidget
{
    Q_OBJECT
    vector<LyricLine> lines;          //储存所有歌词
public:
    explicit LyricWidget(QWidget *parent = nullptr);
    ~LyricWidget();
    bool process(QString filePath); //将歌词文件的内容处理为歌词结构的QList
    int getIndex(qint64 position);  //根据时间找到对应位置的歌词
    void show(qint64 position);     //显示当前播放进度的歌词
    QString getLyricText(int index);//根据下标获得歌词内容
    void clear();                   //清空歌词Label
private:
    Ui::LyricWidget *ui;
};
```

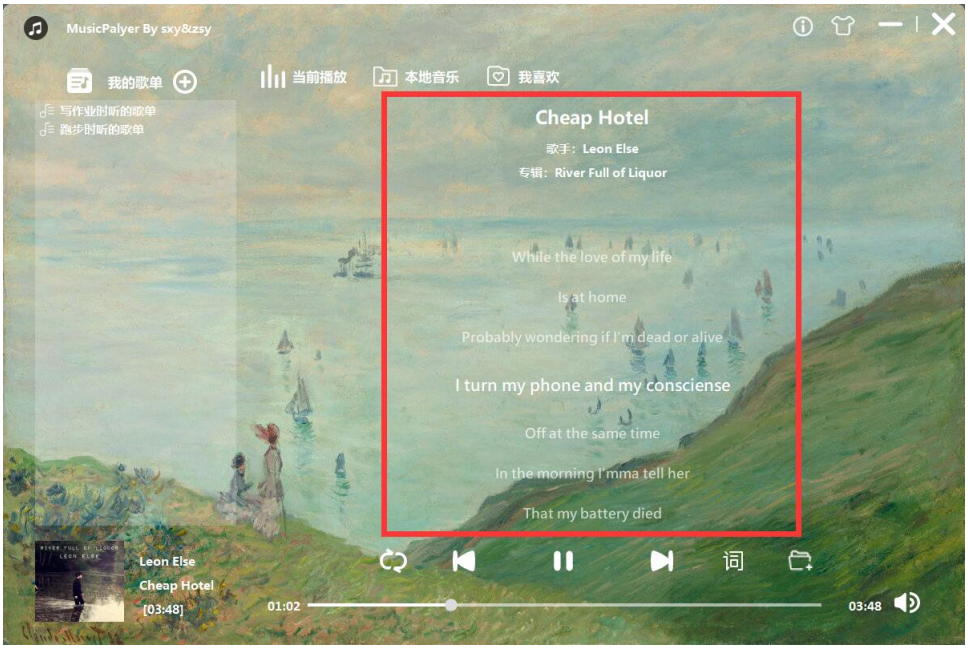
2.2.6 MainWidget 类（主窗口）

> 负责整个播放器各模块的协调与关联，以及承接主页面上各种按钮对应的信号槽，以此实现对应的功能

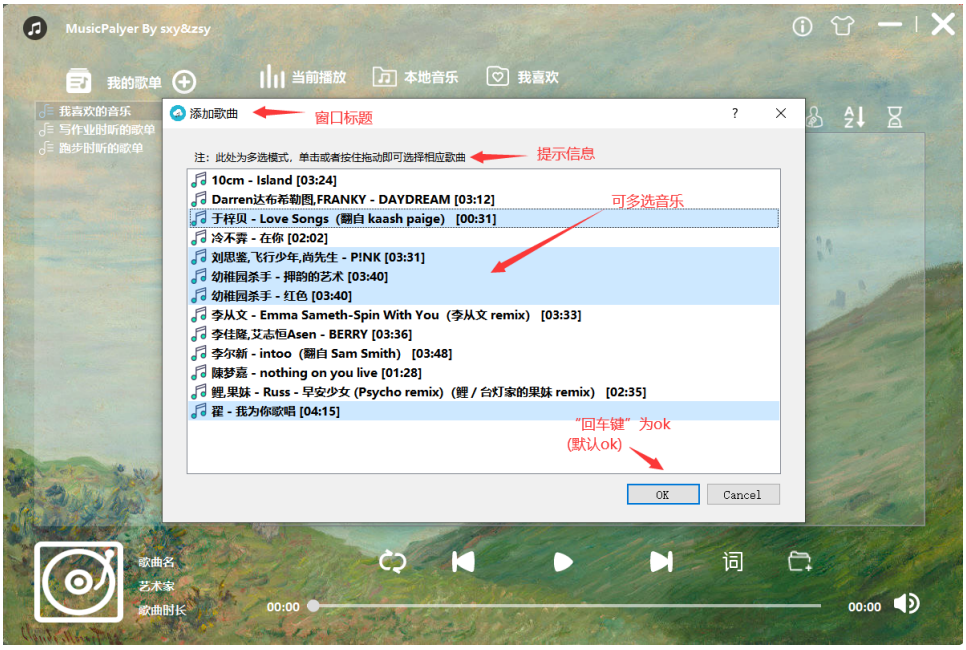
3 界面设计

3.1 主页面

3.2 歌词界面



3.3 ”添加音乐至歌单“界面



4 功能实现简述

5 亮点和小结

5.1 代码亮点简述

以下简述一下本项目的一些亮点（仅代表小组两人观点）：

1. ……
2. ……
3. ……
4.

5.2 项目小结/心得