**第三课**

1. 后台 歌曲管理domain和mapper层

先编写domain 层song.java引入歌曲信息，在编写dao层songMapper歌曲引入所需接口，最后编写resources文件下的mapper文件SongMapper.xml文件编写SQL代码。代码基本和歌手管理的代码一样。

<result column="lyric" property="lyric" jdbcType="LONGVARCHAR" />

LONGVARCHAR需要的时候查询不需要的时候不查。

1. 后台 歌曲管理service

接口和实现类

编写Songservice 和Mapper层接口差不多

实现类在impl 文件下Songserviceimpl

1. 后台 歌曲管理controller层添加方法
2. 后台 歌曲管理添加歌曲

前端SongPage.vue代码和歌手管理类似

<i class=”el-icon-tickes”></i>歌曲信息

在顶端添加一个歌曲信息的小图片

他采用div的提交方法我们采用的是el

他写的代码和我们的不太一样但是功能都差不多。

// req.readyState === 4 获取到返回的完整数据

// req.status === 200 和后台交互完成

1. 后台 歌曲管理 查询

在后端Songcontroller添加根据歌手ID查询歌曲

// 返回指定歌手ID的歌曲

// 根据歌手ID查询歌曲

之后在前端api下的index.ts 添加查询接口

 getSongOfSingerId: (id) => get(`song/singer/detail?singerId=${id}`),

最后在前端SongPage.vue添加查询代码

/ 获取歌曲

    async function getData() {

      tableData.value = [];

      tempDate.value = [];

      const result = (await HttpManager.getSongOfSingerId(singerId.value)) as ResponseBody;

      tableData.value = result.data;

      tempDate.value = result.data;

      currentPage.value = 1;//默认当前页为1

    }

1. 后台 歌曲管理 修改和删除

在后端Songcontroller

在index.ts 里面添加接口

// 删除歌曲

  deleteSong: (id) => get(`song/delete?id=${id}`),

在前端SongPage.vue添加修改和删除代码

1. 后台 歌曲管理 歌词显示

解析歌词代码没找到，但是我们的项目里面是做了的

parseLyric(text){

Let lines = text.split(“\n”)

Let pattern =

/\[\d{2}.\d{3}|d{2}\]/g;