**需求分析文档**

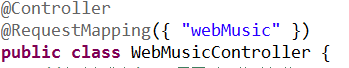
1. **系统前台功能概述**
2. **系统后台功能概述**

**详细设计文档**

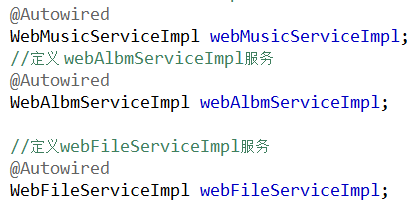
1. **Controller——WebMusicController.java**

**功能：**歌曲信息管理的控制器

1. 将class定义为一个controller类，注解用以定义URL 请求和Controller 方法之间的映射，在类前面定义，将url:webMusic和WebMusicController绑定。



1. 定义服务，便于调用歌手管理，专辑管理、下载管理里面的方法



1. **删除歌曲-- deleteWebAlbm（）**
2. 设置想要跳转的父路径



1. 创建一个result对象



1. 通过music\_id， 调用webMusicServiceImpl接口的删除方法，并且进行判断

**if**(webMusicServiceImpl.deleteWebMusicByMusicId(music\_id)){

1. 如果webMusicServiceImpl接口返回的结果为1，则成功删除歌曲



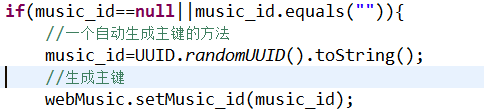
1. 如果webMusicServiceImpl接口的删除方法返回结果为-1，则删除歌曲失败



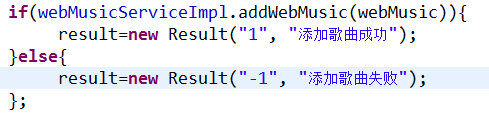
1. **保存歌曲添加或者修改—— WebSingerAddOrUpdateSubmit（）**
2. 定义一个music\_id,并且从歌曲实体类中获取一个music\_id



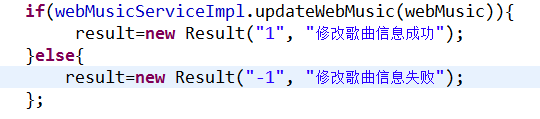
1. 对music\_id进行判断，如果为空，就自动将music\_id生成主键



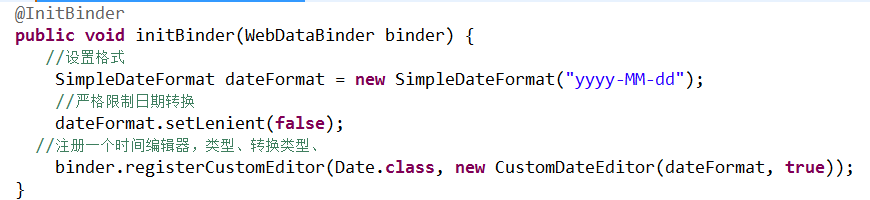
1. 进行添加歌曲操作
2. 调用webMusicServiceImpl接口的添加方法，并且进行判断



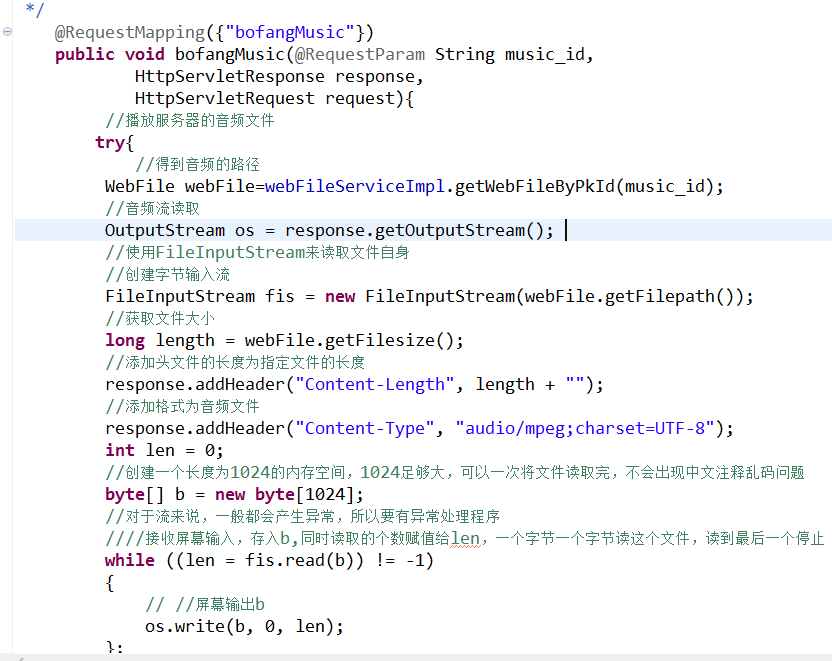
1. 如果返回结果为1，则添加歌曲成功
2. 如果返回结果为-1，则添加歌曲失败
3. 进行修改歌曲操作
4. 调用webMusicServiceImpl接口的修改方法，并且进行判断



1. 如果返回结果为1，则修改歌曲信息成功
2. 如果返回结果为-1，则修改歌曲信息失败
3. **转换时间——**initBinder（）
4. 时间属性的编辑器，时间类型直接会报错



1. 设置格式
2. 严格限制时间转换
3. 注册一个时间编辑器
4. **播放音乐——**bofangMusic（）
5. 播放服务器的音频文件
6. 得到音频的路径
7. 音频流读取
8. 使用FileInputStream来读取文件自身
9. 创建字节输入流
10. 获取文件大小
11. 添加头文件的长度为指定文件的长度
12. 添加格式为音频文件
13. 异常处理
14. 创建一个长度为1024的内存空间，1024足够大，可以一次将文件读取完，不会出现中文注释乱码问题
15. 接收屏幕输入，存入b,同时读取的个数赋值给len，一个字节一个字节读这个文件，读到最后一个停止
16. 屏幕输出b
17. 关闭输入流
18. 关闭输出流



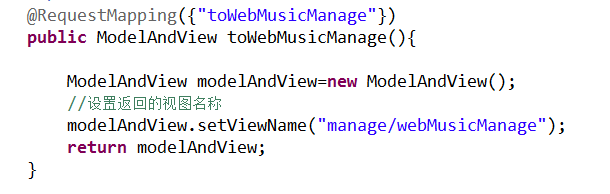
1. **下载歌曲——downloadMusic（）**
2. 得到音频的路径
3. 将该文件加入到输入流之中
4. 设置缓存
5. 返回下一次对此输入流调用的方法可以不受阻塞地从此输入流读取（或跳过）的估计剩余字节数
6. 读入到输入流里面
7. 防止中文乱码，字节流读取String数据
8. 设置请求状态
9. 设置响应码
10. 返回文件给前端



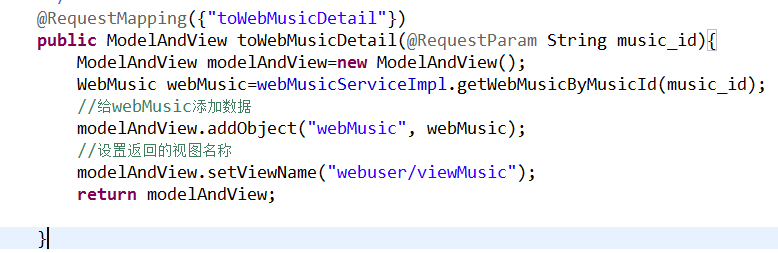
1. **添加或者修改歌曲界面——toWebMusicAddOrUpdate（）**
2. 获取所有专辑列表
3. 定义实体类
4. 定义一个新视图
5. 传入形参
6. 根据music\_id获取WebMusic对象
7. 向ModelAndView中添加WebMusic对象数据
8. 向ModelAndView中添加WebAlbms对象数据，添加模型数据 可以是任意的POJO对象
9. 设置返回的视图名称，返回的文件名
10. 访问jsp文件



1. **后台播放歌曲主界面——toWebMusicManage（）**
2. 返回到manage/webMusicManage
3. 通过ModelAndView构造方法可以指定返回的页面名称，
4. 也可以通过setViewName()方法跳转到指定的页面 ,
5. addObject()有几个不同参数的方法，可以默认和指定返回对象的名字。



1. **用户播放界面——toWebMusicDetail()**
2. 通过ModelAndView构造方法可以指定返回的页面名称
3. 给webMusic添加数据
4. 设置返回的视图名称

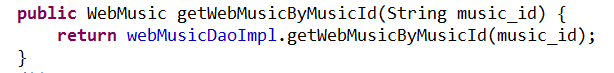


1. **Service--WebMusicService.java**

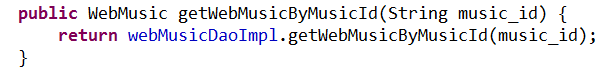
**Service--WebMusicServiceImpl.java**

**功能： 歌曲信息管理的业务逻辑层**

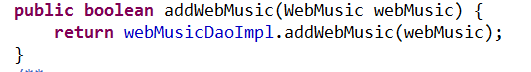
1. **删除歌曲信息——deleteWebMusicByMusicId（）**
2. 通过调用DAO层webMusicDaoImpl里面的删除方法，并且使用music\_id来判断到底删除哪一首歌的信息



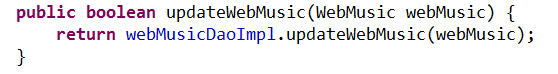
1. **得到歌曲信息——getWebMusicByMusicId（）**
2. 通过调用DAO层webMusicDaoImpl里面的查找方法，并且使用music\_id来判断到底查找哪一首歌的信息



1. **添加歌曲信息——addWebMusic（）**
2. 通过调用DAO层webMusicDaoImpl里面的添加方法，并且使用music\_id来判断到底是否有这首歌，再进行添加操作



1. **修改歌曲信息—— updateWebMusic（）**
2. 通过调用DAO层webMusicDaoImpl里面的修改方法，并且使用music\_id来判断到底修改哪一首一首歌的信息

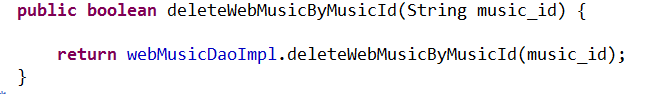


1. **Dao层——WebMusicDao.java**

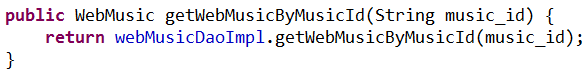
**Dao层——WebMusicDaoImpl.java**

**功能：**通过Dao层封装实体类对象，进行数据库操作

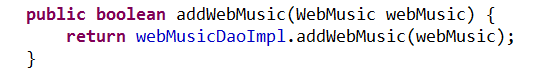
1. 删除歌曲的信息——deleteWebMusicByMusicId（）
2. 通过music\_id，去数据库中找到这首歌
3. 用sqlSessionTemplate去操作数据库 ，删除歌曲的信息



1. 查询歌曲的信息——getWebMusicByMusicId（）
2. 通过music\_id，去数据库中找到这首歌
3. 用sqlSessionTemplate去操作数据库 ，查询歌曲的信息



1. 添加歌曲的信息——addWebMusic（）
2. 通过调用WebMusic中的方法体和参数
3. 用sqlSessionTemplate去操作数据库 ，添加歌曲的信息



1. 修改歌曲的信息——updateWebMusic（）
2. 通过调用WebMusic中的方法体和参数
3. 用sqlSessionTemplate去操作数据库 ，修改歌曲的信息



**四、实体类**

**数据库设计文档**