**详细设计文档**

**一、歌手信息管理的控制器**

**----WebSingerController.java**

**①访问webSinger时跳转到WebSingerController管理器上来**

**public** **class** WebSingerController {

//自动帮你把bean里面引用的对象的setter/getter方法省略，它会自动帮你set/get

//引入webSingerServiceImpl和webFileServiceImpl

@Autowired

WebSingerServiceImpl webSingerServiceImpl;

@Autowired

WebFileServiceImpl webFileServiceImpl;

**原理：**在SpringMVC 中，控制器Controller 负责处理由DispatcherServlet 分发的请求，它把用户请求的数据经过业务处理层处理之后封装成一个Model ，然后再把该Model 返回给对应的View 进行展示；

在SpringMVC 中提供了一个非常简便的定义Controller 的方法，你无需继承特定的类或实现特定的接口；

只需使用@Controller 标记一个类是Controller ，然后使用@RequestMapping 和@RequestParam就能被外界访问到。

**②删除歌手界面deleteWebSinger**

@RequestMapping({"deleteWebSinger"})

//一般在异步获取数据时使用，在使用@RequestMapping后，返回值通常解析为跳转路径

//加上@responsebody后返回结果不会被解析为跳转路径，而是直接写入HTTP response body中。

@ResponseBody

////从@RequestParam中提取形参

1) //创建一个实体类

2) //判断获取到的歌手id是否为空

3) //不为空删除成功，返回返回码1

4) //为空不成功，返回返回码-1

5) //返回result

**public** Result deleteWebSinger(@RequestParam String singer\_id){

Result result=**null**;

**if**(webSingerServiceImpl.deleteWebSingerBySingerId(singer\_id)){

result=**new** Result("1", "删除歌手信息成功");

}**else**{

result=**new** Result("-1", "删除歌手信息失败");

}

**return** result;

}

**③查看歌手信息viewWebSinger**

@RequestMapping({"viewWebSinger"})

//使用ModelAndView类用来存储处理完后的结果数据，以及显示该数据的视图

//ModelAndView中的Model代表模型，View代表视图

1. 实例化一个空websinger
2. 实例化一个ModelAndView类zzzzzzzzzzzzxzz
3. 将获取的歌手id放入websinger中
4. //将歌手信息添加到类中
5. //创建这个名字
6. //返回ModelAndView类

**public** ModelAndView viewWebSinger(HttpServletRequest req,

HttpServletResponse resp,@RequestParam String singer\_id){

WebSinger webSinger=**null**;

ModelAndView modelAndView=**new** ModelAndView();

webSinger=webSingerServiceImpl.getWebSingerBySingerId(singer\_id);

modelAndView.addObject("webSinger", webSinger);

modelAndView.setViewName("manage/webSingerView");

**return** modelAndView;

}

**④歌手新增或者修改界面toWebSingerAddOrUpdate**

@RequestMapping({"toWebSingerAddOrUpdate"})

1. //实例化一个空websinger
2. //实例化一个ModelAndView类
3. //判断歌手id不为空
4. //根据singer\_id获取WebSinger对象
5. //在ModelAndView类中添加或更新歌手信息
6. //在ModelAndView类中添加或更新歌手信息
7. //返回ModelAndView

**public** ModelAndView toWebSingerAddOrUpdate(HttpServletRequest req,

HttpServletResponse resp){

WebSinger webSinger=**null**;

ModelAndView modelAndView=**new** ModelAndView();

String singer\_id=req.getParameter("singer\_id");

**if**(singer\_id!=**null**){

webSinger=webSingerServiceImpl.getWebSingerBySingerId(singer\_id);

}

modelAndView.addObject("webSinger", webSinger);

modelAndView.setViewName("manage/webSingerAddOrUpdate");

**return** modelAndView;

}

**⑤提交歌手添加或者更新操作WebSingerAddOrUpdateSubmit**

@RequestMapping({"WebSingerAddOrUpdateSubmit"})

@ResponseBody

1)根据webSinger对象的singer\_id,判断是添加 还是修改

2) 实例化一个result

3) 返回一个新的String类对象singer\_id

4) 判断如果id为空

5) 如果添加的内容不为空

6) 添加成功，返回返回码1

7) 添加失败，返回返回码-1

8) 如果修改的内容不为空

9) 修改成功，返回返回码1

10)修改失败，返回返回码-1

注解：equals 方法是String类从它的超类Object中继承的, 被用来检测两个对象是否相等，即两个对象的内容是否相等，区分大小写；

UUID.randomUUID().toString()是javaJDK提供的一个自动生成主键的方法.

**public** Result WebSingerAddOrUpdateSubmit(HttpServletRequest req,

HttpServletResponse resp,WebSinger webSinger){

Result result=**null**;

String singer\_id=webSinger.getSinger\_id();

**if**(singer\_id==**null**||singer\_id.equals("")){

singer\_id=UUID.*randomUUID*().toString();

webSinger.setSinger\_id(singer\_id);

**if**(webSingerServiceImpl.addWebSinger(webSinger)){

result=**new** Result("1", "添加歌手成功");

}**else**{

result=**new** Result("-1", "添加歌手失败");

};

}**else**{

**if**(webSingerServiceImpl.updateWebSinger(webSinger)){

result=**new** Result("1", "修改歌手信息成功");

}**else**{

result=**new** Result("-1", "修改歌手信息失败");

};

}

**return** result;

}

**⑥跳转到专辑管理界面toWebSingerManage**

//跳转到webUserManage页面

**public** ModelAndView toWebUserManage(){

ModelAndView modelAndView=**new** ModelAndView();

modelAndView.setViewName("manage/webSingerManage");

**return** modelAndView;

}

**⑦跳转到歌手详情页toViewSinger**

@RequestMapping({"toViewSinger"})

1）获取歌手的id放到websinger中

2）通过webSingerServiceImpl获取歌手的id信息

3）创建新的modelAndView类

4)将webSinger、webSinger添加到类中

5）创建视图的名称

6）返回视图

注解：//调用 addobject 方法时，将触发新添加对象的 Init 事件。在表单集中加入表单时，Load 事件在 Init 事件之前发生。

//通过ModelAndView构造方法可以指定返回的页面名称，也可以通过setViewName()方法跳转到指定的页面。

**public** ModelAndView toWebSingerView(@RequestParam String singer\_id){

WebSinger webSinger=webSingerServiceImpl.getWebSingerBySingerId(singer\_id);

List<WebFile> webFilelists=webFileServiceImpl.getWebFileListByPkId(singer\_id);

ModelAndView modelAndView=**new** ModelAndView();

modelAndView.addObject("webSinger",webSinger);

modelAndView.addObject("webFilelists", webSinger);

modelAndView.setViewName("webuser/viewSingerDetail");

**return** modelAndView;

**二、歌手信息管理的Service层**

**----WebSingerService**

**①引入 webSingerDaoImpl**

@Autowired

WebSingerDaoImpl webSingerDaoImpl;

**②声明函数selectAllWebSingerByPage 选择所有歌手页面**

**public** ServiceToActionMsg<WebSinger> selectAllWebSingerByPage(Page<WebSinger> pager);

**③声明函数getWebSingerBySingerId获取歌手id；**

**public** WebSinger getWebSingerBySingerId(String singer\_id);

**④声明函数 addWebSinger 添加歌手**

**public** **boolean** addWebSinger(WebSinger webSinger);

**⑤声明函数updateWebSinger更新歌手信息**

**public** **boolean** updateWebSinger(WebSinger webSinger);

**⑥声明函数deleteWebSingerBySingerId删除歌手信息**

**public** **boolean** deleteWebSingerBySingerId(String singer\_id);

**⑦声明函数 getAllSingers 获取所有歌手信息**

**public** List<WebSinger> getAllSingers();

**三、歌手信息管理ServiceImpl层**

**----WebSingerServiceImpl**

**①声明函数selectAllWebSingerByPage 选择所有歌手页面**

**public** ServiceToActionMsg<WebSinger> selectAllWebSingerByPage(Page<WebSinger> pager);

**②声明函数getWebSingerBySingerId获取歌手id；**

**public** WebSinger getWebSingerBySingerId(String singer\_id);

③**声明函数 addWebSinger 添加歌手**

**public** **boolean** addWebSinger(WebSinger webSinger);

**④声明函数updateWebSinger更新歌手信息**

**public** **boolean** updateWebSinger(WebSinger webSinger);

⑤**声明函数deleteWebSingerBySingerId删除歌手信息**

**public** **boolean** deleteWebSingerBySingerId(String singer\_id);

**⑥声明函数 getAllSingers 获取所有歌手信息**

**public** List<WebSinger> getAllSingers();

**四、歌手信息管理Dao层**

**----WebSingerDao**

**①声明函数selectAllWebSingerByPage 选择所有歌手页面**

**public** Page<WebSinger> selectAllByPage(Page<WebSinger> pager, WebSinger t);

**②声明函数getWebSingerBySingerId获取歌手id；**

**public** WebSinger getWebSingerBySingerId(String singer\_id);

③**声明函数 addWebSinger 添加歌手**

**public** **boolean** addWebSinger(WebSinger webSinger);

**④声明函数updateWebSinger更新歌手信息**

**public** **boolean** updateWebSinger(WebSinger webSinger);

⑤**声明函数deleteWebSingerBySingerId删除歌手信息**

**public** **boolean** deleteWebSingerBySingerId(String singer\_id);

**⑥声明函数 getAllSingers 获取所有歌手信息**

**public** List<WebSinger> getAllSingers();

**五、歌手信息管理Daoimpl层**

**----WebSingerDaoImpl**

**①声明函数selectAllWebSingerByPage 选择所有歌手页面**

**public** Page<WebSinger> selectAllByPage(Page<WebSinger> pager, WebSinger t);

**②声明函数getWebSingerBySingerId获取歌手id；**

**public** WebSinger getWebSingerBySingerId(String singer\_id);

③**声明函数 addWebSinger 添加歌手**

**public** **boolean** addWebSinger(WebSinger webSinger);

**④声明函数updateWebSinger更新歌手信息**

**public** **boolean** updateWebSinger(WebSinger webSinger);

⑤**声明函数deleteWebSingerBySingerId删除歌手信息**

**public** **boolean** deleteWebSingerBySingerId(String singer\_id);

**⑥声明函数 getAllSingers 获取所有歌手信息**

**public** List<WebSinger> getAllSingers();

**六、歌手信息管理Daomain层（model层）**

**----WebSinger**

将singer\_id歌手id**、**singer\_name歌手姓名**、**singer\_birthdate歌手出生日期、singer\_shuxiang歌手属相、singer\_xingzuo歌手星座、singer\_aihao歌手爱好、singer\_jianjie歌手简介封装在模型中，用setsinger——××、getsinger——××来对数据进行设置和获取。