**《Java企业级框架》**

**课程设计报告**

|  |  |  |
| --- | --- | --- |
| 姓名 | ： | 靳飞宇 |
| 班级 | ： | 21网一 |
| 日期 | ： | 二〇二三年六月 |

**一、设计需求**

主要介绍在该课程设计中你所实现的功能，这些功能需要满足哪些要求。

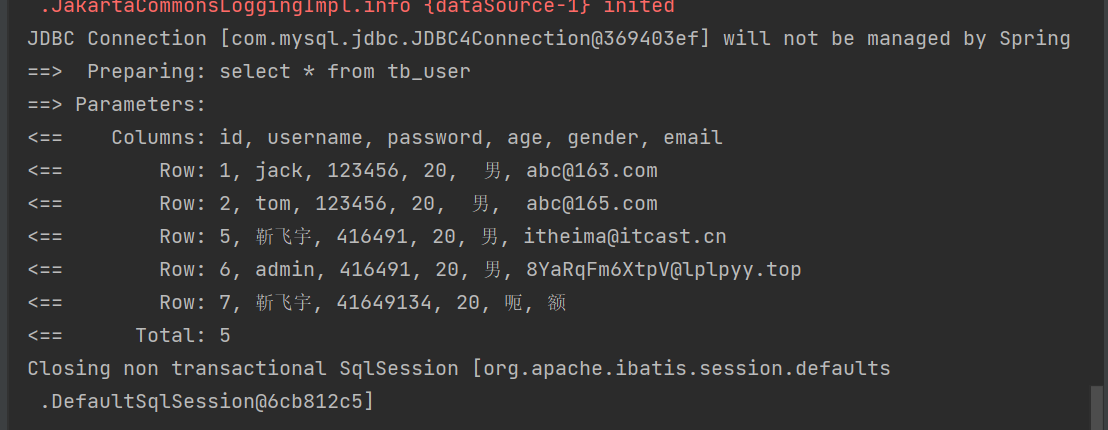
本次课程设计我总共实现了四项功能:

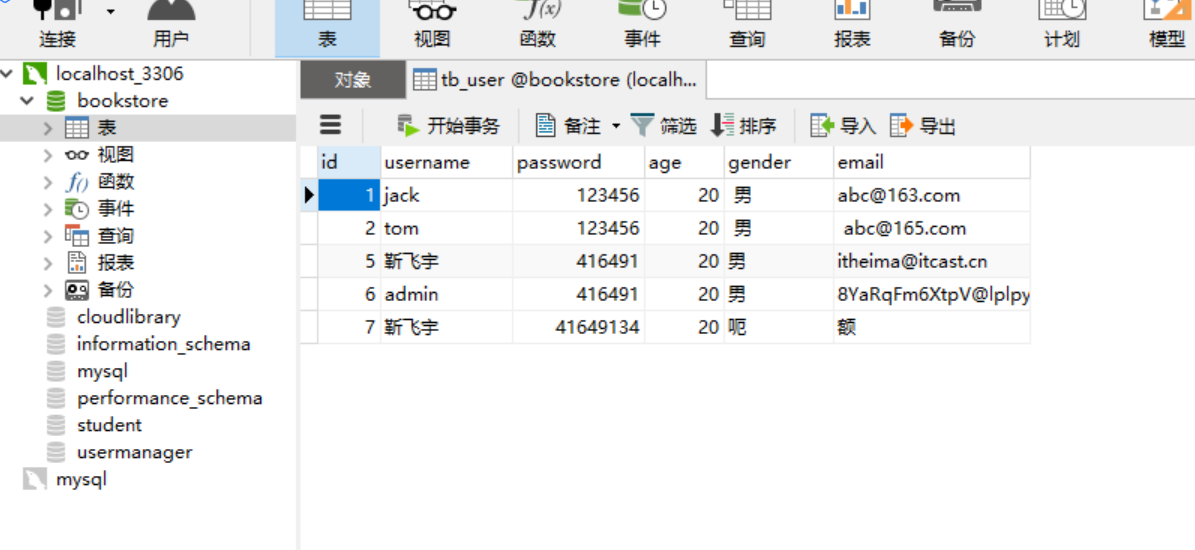
**1,查看所有用户的功能**



当我们点击操作菜单左侧的查看用户的二级列表后右侧会根据前端页面的Ajax请求渲染显示如下图的页面，其中上半部分的输入框用于对从数据库请求的数据进行筛选，如性别的复选框男和女，下半部分是我们从数据库查询的内容

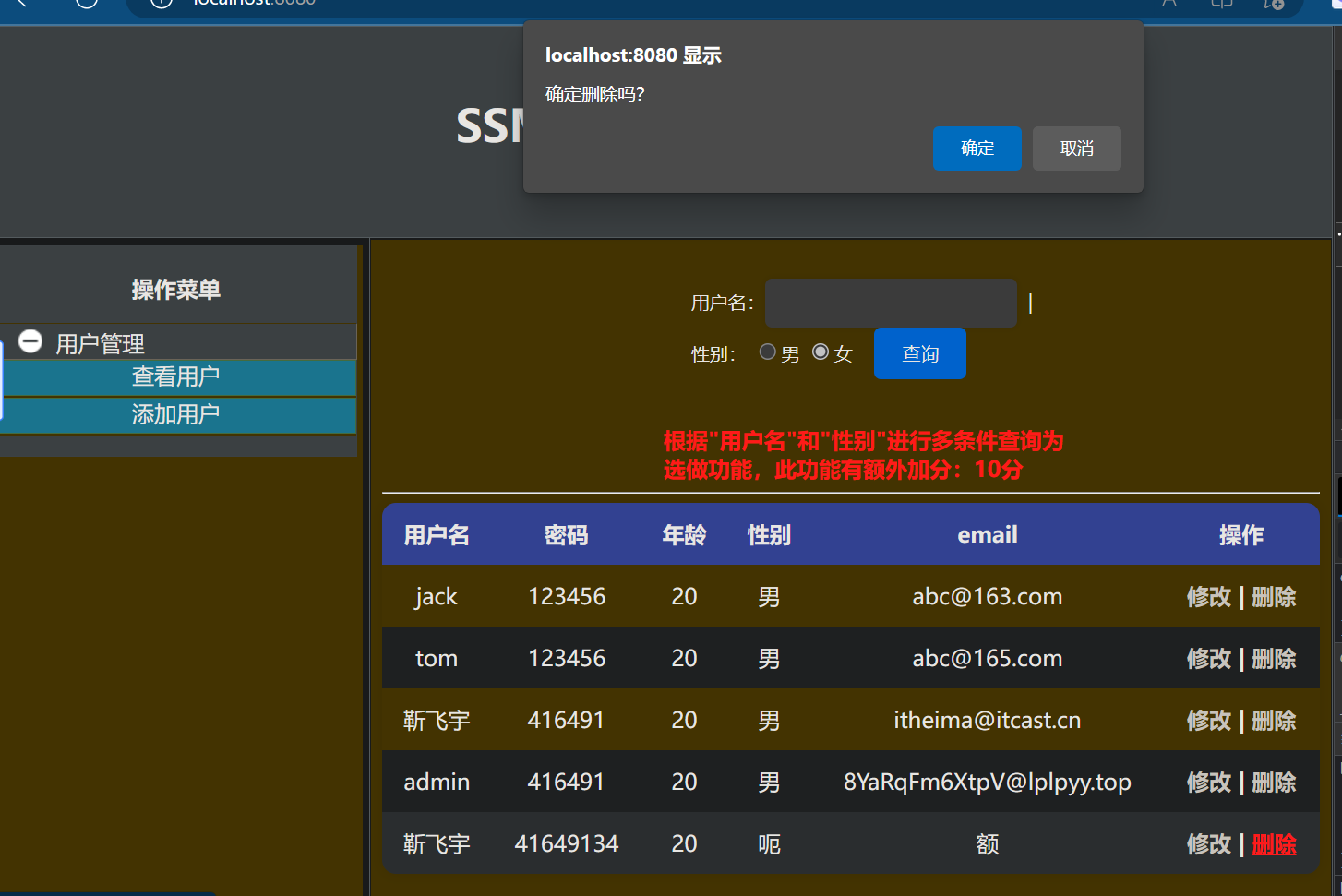
****

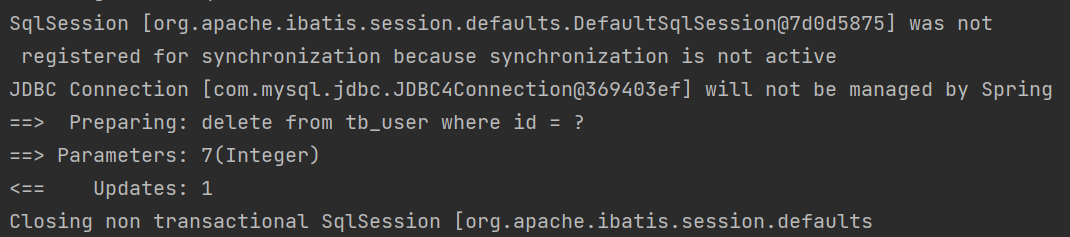
****

**：：：：：：：数据库对照如下：：：：：：：：**

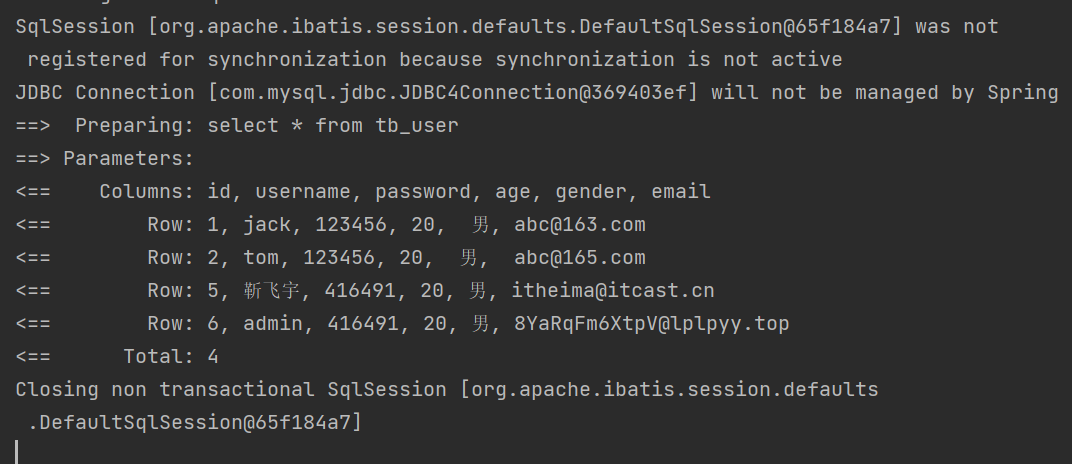
**2,** 用户的修改和删除

(1)用户的删除

****

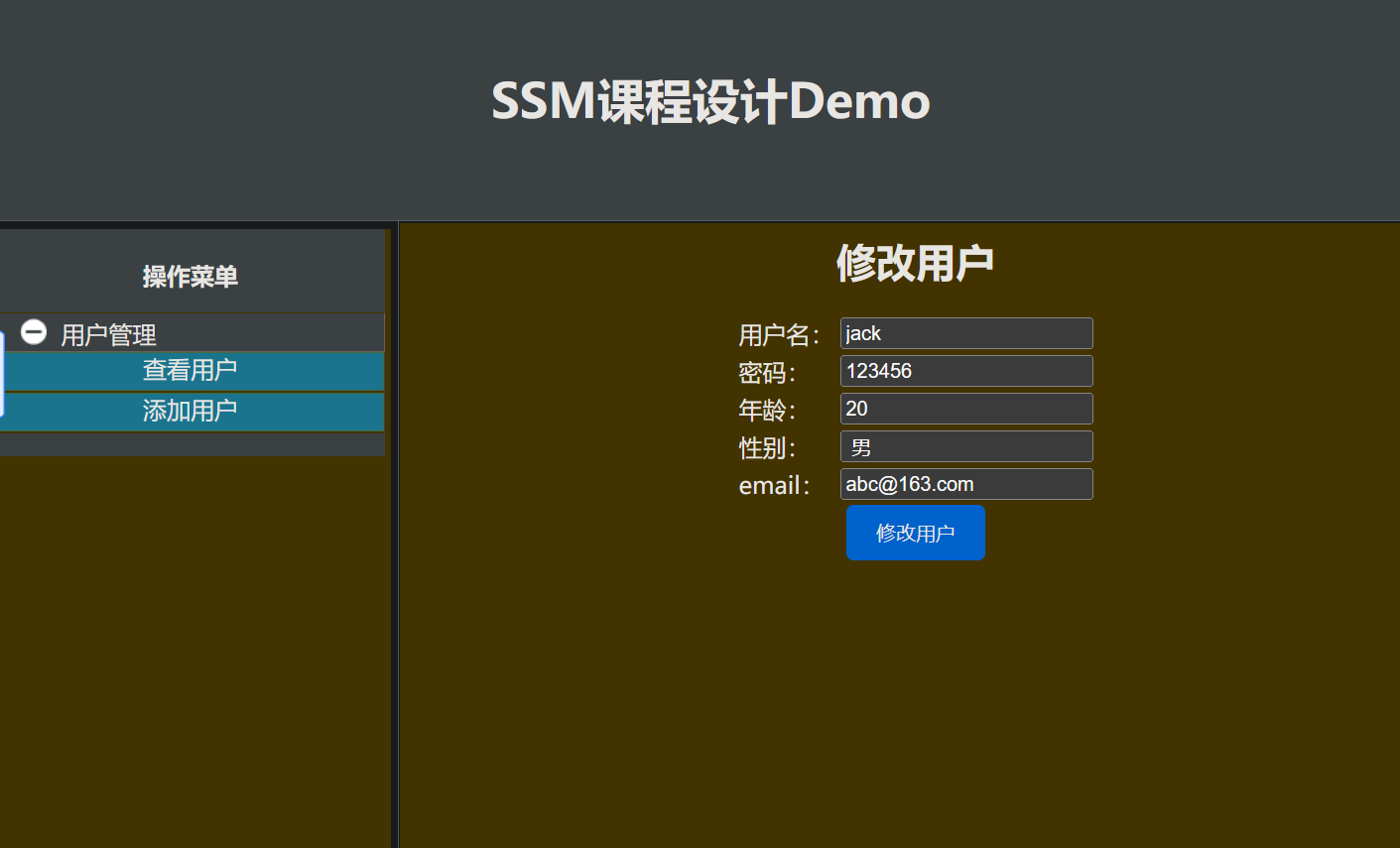
**当我们点击页面右下角的删除超链接时候，页面会弹出复选框如上图提示是否删除该用户。我们点击确定后就会自动删除这一用户。**

**删除之后后端的数据库会再次查询数据库剩余的所有用户。**

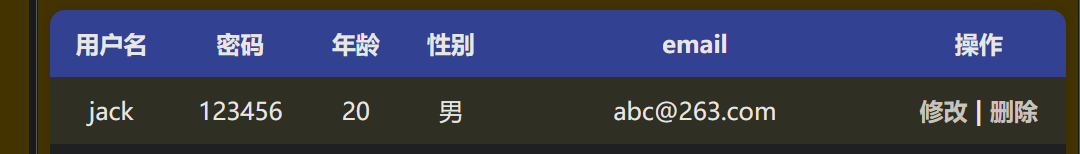
****

**（2）用户的修改**

**如图点击修改超链接页面会跳转到我们的表单修改页面，也就对应modify.jsp**

****

**用户名的输入框是锁定的，他是不能被修改的，这也保证了修改用户不能修改它的名字，其他的部分可以任意修改**

****

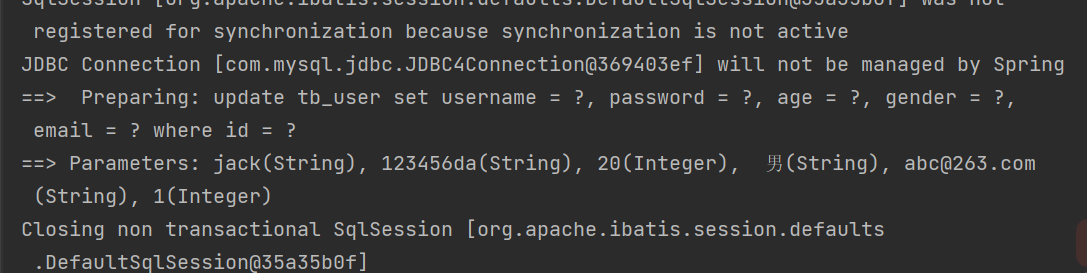
如把上图的jack邮箱改变

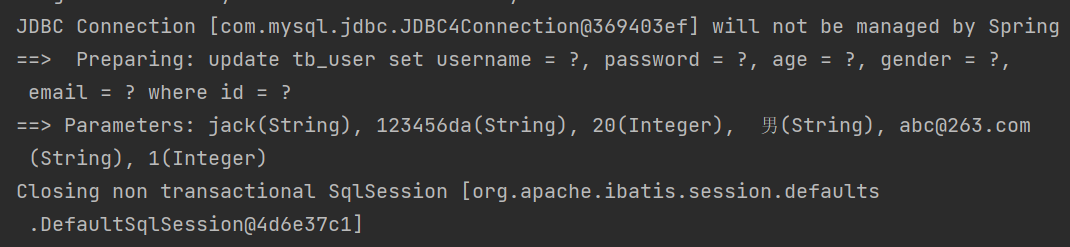
**3，添加用户功能**

****

**点击添加用户的功能后会弹到我们添加用户的页面,输入后点击添加用户**

****

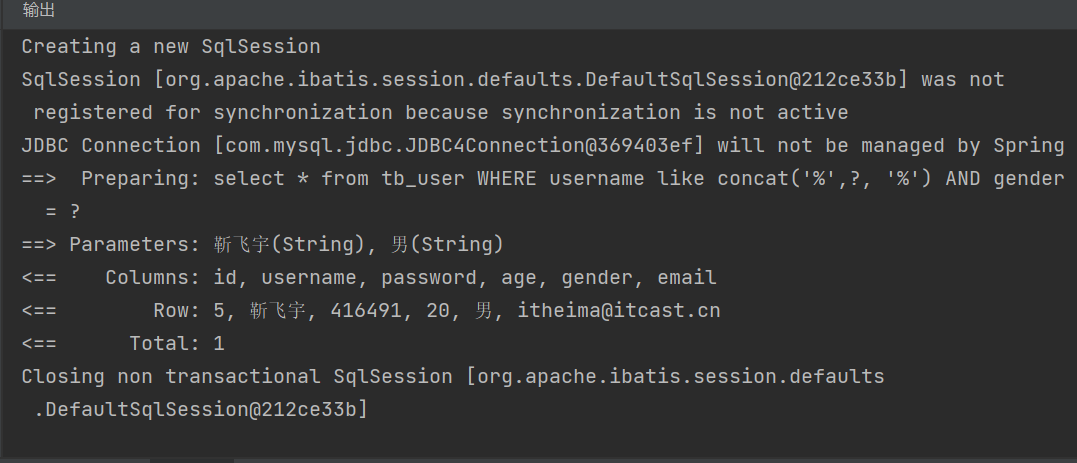
****

****

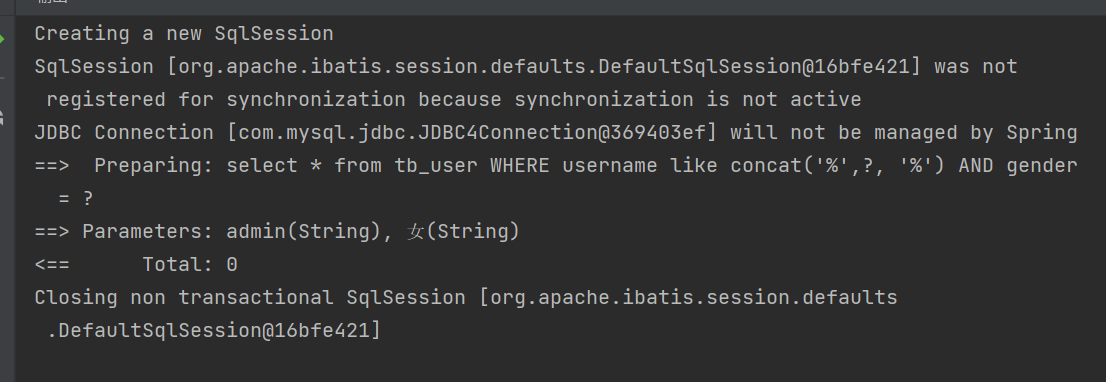
****

**可以看见已经添加到我们的表格里，后端也通过数据库的更新语句进行更新。**

**4，多条件查询**

**当输入数据库中存在的人的名字和性别时候，后端数据库会执行一个名字的模糊查询和性别的精确查询从而查询到我们的数据库中的用户**

**当输入不存在的人，则查询不到**

****

****

**二、总体设计**

该课程设计虽然只涉及到用户管理模块，也请以“**总体功能**”图来描述该模块的功能。

+------------------------------------------------------+

| 用户管理模块 |

+------------------------------------------------------+

| |

| 查看所有用户 |

| （列表显示所有用户） |

| |

+------------------------------------------------------+

| |

| 添加用户功能 |

| （输入用户信息并保存） |

| |

+------------------------------------------------------+

| |

| 修改用户功能 |

| （选择用户并修改信息） |

| |

+------------------------------------------------------+

| |

| 删除用户功能 |

| （选择用户并删除） |

| |

+------------------------------------------------------+

| |

| 多条件查询功能 |

| （按条件查询用户列表） |

| |

+------------------------------------------------------+

该课程设计你采用什么样的架构进行开发设计，有哪些核心类（接口），对其作用进行说明（可以使用图形或文字进行说明）。

在该课程设计中，我采用了经典的MVC（Model-View-Controller）架构进行开发设计。这种架构将系统分为三个主要组件：模型（Model）、视图（View）和控制器（Controller），它们各自负责不同的职责。

1. 模型（Model）：模型层表示应用程序的数据和业务逻辑。在用户管理模块中，核心的模型类是User，它表示系统中的用户对象。模型类通常包含数据的定义和访问方法，以及与数据库进行交互的操作。模型层的主要作用是封装数据和业务逻辑，提供对数据的增删改查等操作。

2. 视图（View）：视图层负责用户界面的展示和用户交互。在用户管理模块中，视图层包括了用于显示用户列表、添加用户、修改用户和查询用户的页面。视图层通过向用户展示数据，并接收用户的输入来实现与用户的交互。视图层通常使用HTML、CSS和JavaScript等前端技术进行开发。

3. 控制器（Controller）：控制器层负责处理用户请求、协调模型和视图之间的交互。在用户管理模块中，核心的控制器类是UserController，它接收用户的请求，根据请求的类型调用相应的模型操作，并将结果传递给适当的视图进行展示。控制器层的主要作用是实现业务逻辑的控制和协调，将用户的请求转化为相应的操作。

下图是一个简化的MVC架构示意图：

```

+---------------------+

| View |

| |

| HTML/CSS/JS |

| |

+----------^----------+

|

| 用户交互

|

+----------+----------+

| Controller |

| |

| UserController |

| |

+----------^----------+

|

| 操作模型

|

+----------+----------+

| Model |

| |

| UserDAO |

| |

+---------------------+

```

在这个架构中，用户通过视图层与系统进行交互，视图层将用户的请求传递给控制器层。控制器层根据请求的类型调用相应的模型操作，获取数据并将结果传递给视图层进行展示。

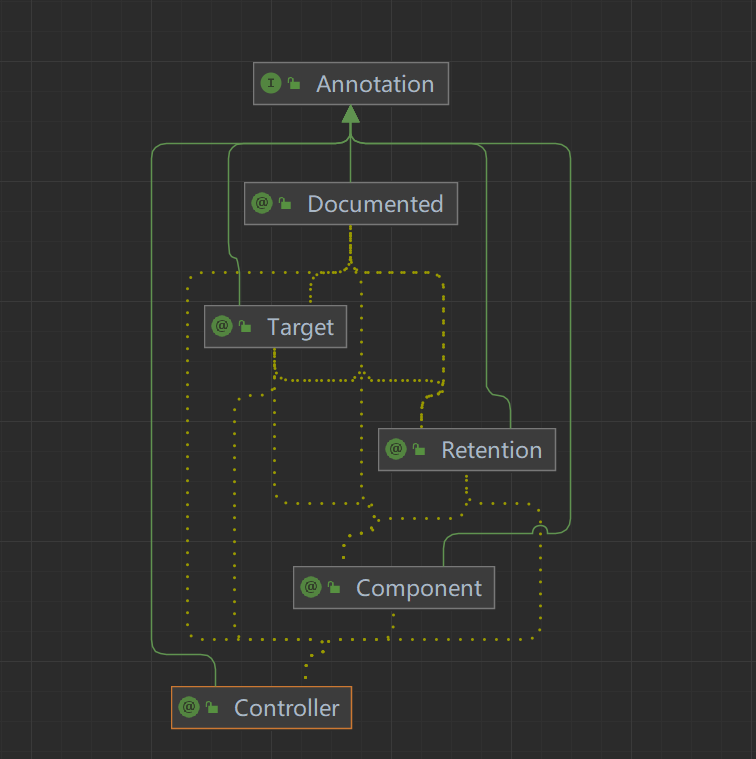
在课程设计中，可能还会涉及其他的核心类和接口，例如数据访问对象（DAO）、服务层（Service）、数据传输对象（DTO）等，这取决于系统的需求和设计决策。这些核心类和接口的作用是实现系统的具体业务逻辑和数据访问操作，通过彼此之间的协作完成系统的功能。

1. **详细设计**

根据自己的设计思路与想要实现的功能，编写主要功能代码。此部分可以粘贴一些控制层或业务层的一些相关代码，因为前端页面已经提供好，关于页面的代码可以不用涉及。

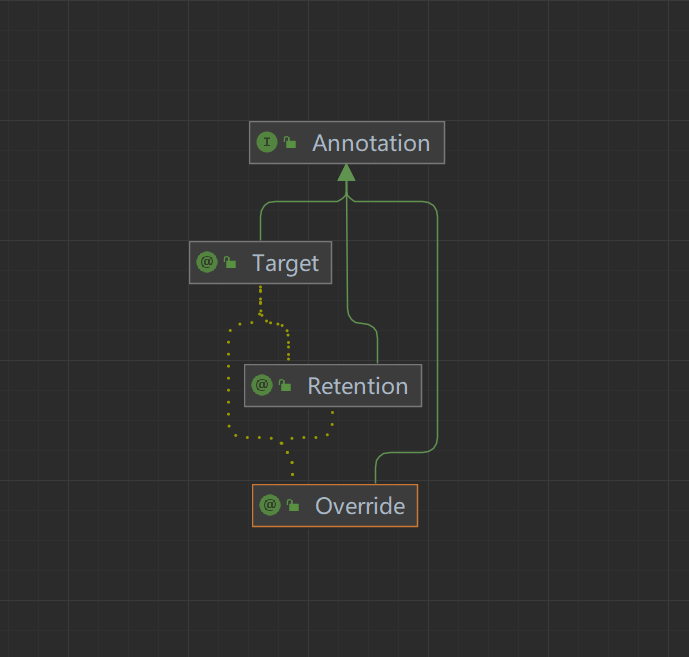
不要直接将代码粘贴到Word中，那样显得非常杂乱，可以放到文本框中，下面是一个示例：

**UserController**



@Controller  
public class UserController {  
 @Resource  
 private UserService userServiceImpl;  
 @GetMapping("/user")  
 public ModelAndView userList(User user) {  
 ModelAndView modelAndView = new ModelAndView("jsp/user/list");  
 List<User> users = userServiceImpl.selectUserAll(user);  
  
 modelAndView.addObject("users", users);  
 return modelAndView;  
 }  
 @GetMapping("/user/modify")  
 public ModelAndView modify(int id) {  
 ModelAndView modelAndView = new ModelAndView("jsp/user/modify");  
 User user = userServiceImpl.selectUserById(id);  
 modelAndView.addObject("user", user);  
 return modelAndView;  
 }  
 @PostMapping("/user/modify")  
 public String modifyPost(User user) {  
 userServiceImpl.updateUser(user);  
 return "redirect:/user";  
 }  
 @GetMapping("/user/del")  
 public String deleteUser(int id) {  
 userServiceImpl.deleteUser(id);  
 return "redirect:/user";  
 }  
 @PostMapping("/user/addUser")  
 public String addUser(User user) {  
 int i = userServiceImpl.insertUser(user);  
 return "redirect:/user";  
 }  
}

**四、部署及运行**



UserServiceImpl.java

@Service  
public class UserServiceImpl implements UserService {  
 @Resource  
 private UserMapper userMapper;  
 @Override  
 public List<User> selectUserAll(User user) {  
 return userMapper.selectUserAll(user);  
 }  
 @Override  
 public int updateUser(User user) {  
 return userMapper.updateUser(user);  
 }  
 @Override  
 public int insertUser(User user) {  
 return userMapper.insertUser(user);  
 }  
 @Override  
 public int deleteUser(int id) {  
 return userMapper.deleteUser(id);  
 }  
 @Override  
 public User selectUserById(int id) {  
 return userMapper.selectUserById(id);  
 }  
}

将设计好的项目部署到Tomcat中运行，首先保证能运行起来，通过浏览器访问应用，验证所有功能是否实现。

**五、设计体会**

UserController 是控制器类，负责处理用户管理模块的请求。UserService 是服务类，负责处理用户相关的业务逻辑。

selectUserAll() 方法用于获取所有用户列表。

selectUserById() 方法根据用户ID获取特定用户的详细信息。

insertUser() 方法用于添加用户。

updateUser() 方法用于修改用户信息。

deleteUser() 方法用于删除用户。

这些方法使用了Spring MVC的注解来定义请求映射和请求方法，通过调用UserService中的对应方法来实现具体的业务逻辑。

接口类UserService

public interface UserService {  
 List<User> selectUserAll(User user);  
 int updateUser(User user);  
 int insertUser(User user);  
 int deleteUser(int id);  
 User selectUserById(int id);  
}  
 }  
 @Override  
 public int deleteUser(int id) {  
 return userMapper.deleteUser(id);  
 }  
 @Override  
 public User selectUserById(int id) {  
 return userMapper.selectUserById(id);  
 }  
}

在设计过程中遇到的问题和困难，如何解决的；此次课程设计中你觉得哪一部分做的是比较满意的或者哪些地方有自己比较新颖的东西；此次课程设计的收获及感想等。（此部分不少于300字，且不能相互抄袭，否则严惩。）

在设计过程中，我遇到了一些挑战和困难，但通过努力和解决方案，我成功地克服了它们。一个具体的问题是如何处理表单验证。我需要确保用户输入的数据符合要求，并在提交前进行验证。我克服这个问题的方式是使用前端技术，例如JavaScript和HTML5表单验证功能，对用户输入进行验证并给予相应的提示。这样，我能够有效地控制用户输入的数据，并提供友好的用户体验。

在整个课程设计中，我对整体的架构设计感到满意。采用了经典的三层架构，使代码结构清晰，提高了代码的可维护性和可扩展性。通过将业务逻辑、数据访问和表示层分离，我能够更好地管理和组织代码，提高开发效率和代码的可读性。

在课程设计中，我还尝试了一些新颖的东西。例如，我使用弹出对话框来确认用户的操作，这样可以增加用户的交互体验和操作的安全性。我还对前端页面进行了样式和布局的优化，使页面更加美观和易用。这些创新点让我的设计与众不同，为用户提供了更好的使用体验。

通过这次课程设计，我获得了宝贵的经验和收获。我学会了如何使用Java企业级框架进行开发，并掌握了三层架构的设计原则和实践方法。我还提升了自己的问题解决能力和学习能力，在遇到困难时能够主动寻求解决方案，并不断学习和进步。

总的来说，这次课程设计是一次有意义的经历。通过设计和实现一个完整的功能模块，我深入理解了软件开发的过程和方法，提高了自己的实际操作能力。我相信这些经验和收获对我的未来发展将有很大的帮助，我会继续努力学习和进步，成为一名优秀的软件开发者。