**课程设计报告**

课程名称 Java MySQL课程设计

设计题目 学生宿舍管理系统的设计与实现

制作时间 \_\_\_\_\_\_\_\_\_\_—\_\_\_\_\_\_\_\_\_\_

姓 名 \_\_\_\_\_\_\_\_\_\_

专业班级 \_\_\_\_\_\_\_\_\_\_

学 号 \_\_\_\_\_\_\_\_\_\_

指导老师 \_\_\_\_\_\_\_\_\_\_

目录

[1 系统分析与设计 4](#_Toc256000000)

[1.1 摘要 4](#_Toc256000001)

[1.2 关键词 6](#_Toc256000002)

[2 引言 7](#_Toc256000003)

[2.1 系统概况 7](#_Toc256000004)

[2.2 系统背景和建设意义 8](#_Toc256000005)

[2.3 目的与分析 9](#_Toc256000006)

[3 系统需求分析 11](#_Toc256000007)

[3.1 功能需求分析 11](#_Toc256000008)

[3.2 非功能需求分析 13](#_Toc256000009)

[4 相关技术介绍 15](#_Toc256000010)

[5 系统开发平台及运行环境 19](#_Toc256000011)

[6 总体设计和业务流程 21](#_Toc256000012)

[7 数据库设计 23](#_Toc256000013)

[7.1 概念设计 23](#_Toc256000014)

[7.2 表结构设计 29](#_Toc256000015)

[8 功能实现 32](#_Toc256000016)

[8.1 概要设计 32](#_Toc256000017)

[8.2 项目截图 34](#_Toc256000018)

[8.3 代码实现 35](#_Toc256000019)

[8.3.1 用户模块 35](#_Toc256000020)

[8.3.2 公告模块 56](#_Toc256000021)

[8.3.3 宿管员模块 74](#_Toc256000022)

[8.3.4 宿舍模块 93](#_Toc256000023)

[8.3.5 学生模块 113](#_Toc256000024)

[9 总结展望 135](#_Toc256000025)

[10 参考文献 137](#_Toc256000026)

# 1 系统分析与设计

## 1.1 摘要

伴随着信息技术高速地发展布局以及网络的快速普及学生宿舍管理系统在当下正在逐渐发挥着越来越重要的作用。在这个时代,能够开发一个一个功能完善,界面友好的基于JavaWEB的学生宿舍管理系统,对我们社会的发展来说是很重要的。

随着计算机技术的不断推进与发展，计算机技术所扮演的角色在学生宿舍信息管理领域中的重要性也逐步显现。就目前的学生宿舍信息管理来讲，传统的手工操作方式已不能适应现代学生宿舍管理的需求。因此研究开发具有高效性和实用性的学生宿舍管理系统已成为必然趋势。

现如今，学生宿舍管理相对于以前已经有了很大的改善，但是随着经济技术的不断发展，功能也需求逐渐增加，为了给用户和信息使用者带来更好的体验，因此就需要设计一个更加完善的学生宿舍管理系统来实现对这些数据的有效管理。本系统的主要任务就是对用户信息、公告信息、宿管员信息、宿舍信息、学生信息进行统一的管理，满足使用者的各类需求。

本系统主要实现了对学生宿舍相关信息的录入到学生宿舍相关信息的管理功能。其主要描述了系统菜单的设计、用户验证、注册、数据库表的设计、用户管理、公告管理、宿管员管理、宿舍管理、学生管理等一系列操作与功能。整个系统从操作简便、灵活、实用的目的与要求出发完成学生宿舍管理系统的全过程。整个系统实现了用户信息查询、浏览、添加、修改、删除；公告信息查询、浏览、添加、修改、删除；宿管员信息查询、浏览、添加、修改、删除；宿舍信息查询、浏览、添加、修改、删除；学生信息查询、浏览、添加、修改、删除等一系列功能。

学生宿舍管理系统是利用JDK作为开发和编译环境，使用idea作为集成开发工具，使用MySQL作为后台数据存储服务，使用Navicat for MySQL作为MySQL的可视化管理工具，使用Tomcat作为最终的运行环境。管理人员可以通过该系统方便、快速、有效的管理用户、公告、宿管员、宿舍、学生等信息。

该系统基于B/S架构模式，配合功能强大的SQL查询语言实现建立关系数据库，访问数据库，对数据库进行更新维护，较好地实现了包括添加、修改、删除、查询等必要的需求功能。该系统方便对用户信息、公告信息、宿管员信息、宿舍信息、学生信息等进行管理，及时了解各个环节中信息的变化，实现用户信息、公告信息、宿管员信息、宿舍信息、学生信息等管理的自动化。

本系统功能比较完善，界面友好，操作简单，方便用户对用户信息、公告信息、宿管员信息、宿舍信息、学生信息等的管理，同时也根据用户的需求，快捷方便的为用户提供查询浏览等服务。

本文对学生宿舍信息管理的业务流程进行了深度研究与研讨，经过充分了解之后对系统进行设计。充分利用现有计算机相关技术，设计出符合现代学生宿舍的信息化管理系统。主要采用了JSP+Servlet技术和MySQL后台数据库方式，实现B/S模式下的学生宿舍管理系统。按照模块化的设计思想对学生宿舍管理系统进行分层和功能设计。分层设计解决程序冗余的问题，从而提高了系统对数据访问和存储的效率；模块化设计遵循“高内聚，低耦合”的思想，将系统划分为用用户管理模块、公告管理模块、宿管员管理模块、宿舍管理模块、学生管理模块等。经实践验证分析，该系统能够在一定程度上提高学生宿舍相关业务处理效率，且能够及时有效地处理繁琐的业务功能，对现代化信息管理与发展起到推进作用。

## 1.2 关键词

学生宿舍管理系统 系统功能 学生宿舍 设计 javaweb JSP 数据库

# 2 引言

## 2.1 系统概况

在现代信息化高速发展的今天，任何工作都离不开信息化和电子化，为了更好地提高工作效率，面对繁杂和庞大的学生宿舍相关信息，我们开发了学生宿舍管理系统，更加有效地管理各种学生宿舍相关信息，既降低了出错的概率，也极大方便了使用人员的日常工作。本学生宿舍管理系统采用Web应用开发语言JSP，拥有继承性的跨平台、面向对象、安全的特点。结合了因特网的发展与应用现状，采用基于Web的三层体系架构，即浏览器—Web应用服务器—数据库服务器。通过对系统工作流程进行系统分析、设计，后台数据库采用MySQL，系统通过Java Bean来连接数据库，系统经过调试，运行效果稳定，操作方便快捷。

在传统模式下利用人工进行学生宿舍信息管理，存在着较多的缺点，如：效率底，保密性差，时间一长将产生大量的文件和数据，更不便于查找，更新，维护等。诸如这些情况，令使用者对学生宿舍信息管理带来了很大困难，严重影响了用户者的工作效率。随着科学技术的不断提高，计算机科学日渐成熟， 使用日趋成熟的计算机技术来代替传统的人工模式，来实现学生宿舍相关信息的现代化管理，其强大的功能已为人们深刻认识，它已进入人类社会的各个领域并发挥着越来越重要的作用。作为计算机应用的一部分，使用计算机对学生宿舍信息进行管理，具有着手工管理所无法比拟的优点。例如:检索迅速、查找方便、易修改、可靠性高、存储量大、数据处理快捷、保密性好、寿命长、成本低、便于打印等。这些优点能够极大地提高信息管理的效率，也是更为科学化、正规化的管理，与世界接轨的重要条件。用计算机制作的学生宿舍管理系统还可以通过功能强大的Internet网及时的向用户传递最新信息，有助于用户对信息的掌握和使用，为更好地把握决策方针起到一定的促进作用！因此，开发这样一套管理软件成为很有必要和有意义的事情。

本次设计使用JDK开发环境、idea作为开发工具，后台数据库采用MySQL，页面显示使用JSP，构成了一个完整的学生宿舍管理系统。使得用户方便快捷的管理和查找等一系列的操作。

## 2.2 系统背景和建设意义

设计一个学生宿舍管理系统，由计算机来代替人工执行一系列诸如对用户信息、公告信息、宿管员信息、宿舍信息、学生信息的处理操作，以方便对以上信息的管理组织工作。

建立学生宿舍管理系统，采用计算机对学生宿舍相关信息进行管理，进一步提高工作效益和现代化水平。帮助广大用户提高工作效率，实现学生宿舍信息管理工作流程的系统化、规范化和自动化。

学生宿舍管理系统必定是现代社会不可缺少的部分。一个功能齐全、简单易用的学生宿舍管理系统不但能有效地减轻用户的工作负担，它的内容对于决策者和管理者来说都至关重要。所以学生宿舍管理系统应该能够为用户提供充足的信息和快捷的查询手段。

现阶段，学生宿舍信息管理系统大多采用传统的人工管理模式。在此体系的统筹策划和实际操纵下，若要进行诸项信息管理工作，比如，组织一次信息录入、信息审核校对、定向信息的调入抽取、数据统计和实时查询等等系列过程，要有几个必需的“人工”操作程序步骤，即人工制表、人工信息录入、人工校对、人工调取、人工统计和人工查询等。显然，随着信息类型的不断增加、更新和转换及信息管理要求的不断提高，日益强调“综合型服务”、“高效性办公”、“实时性信息处理”，信息管理人员的工作量将会越来越大，并且诸项工作必将是十分繁琐、极易出错和误差率高的事情，可以说传统的信息管理方式已经不能适应现代科学管理体系的需要。调研表明，虽然不少已经成功运用了计算机领域技术来管理学生宿舍相关工作，但因相关管理人员的技能水平和技术要求均不高，很大程度上，工作者还是利用Microsoft系列office文档办公软件（Microsoft word、Microsoft power point、Microsoft Access及Microsoft Excel等）来完成报表，这样低水平运作的办公方式和办事效率着实给实时信息查询和信息统计造成了诸多不便。

## 2.3 目的与分析

近年来，随着信息时代的到来，学生宿舍相关信息的数量的不断上升，人工式学生宿舍信息管理越发的凸显了其效率低、成本高、管理不规范的缺点，人工式学生宿舍信息管理已经不能够适应时代的发展，正在逐渐的被信息时代所淘汰，因此学生宿舍信息管理的信息化已经成了必然的趋势。如何充分利用已有的资源对庞大的信息做出科学的、规范的管理是使用者要面临的主要问题。

学生宿舍管理系统可以帮助我们完成对信息的有效存储和永不丢失，也能对信息进行有效整理，极大缓解了人员的工作负担。而且能够知道所需要的具体信息，执行效率也是非常方便快捷，几乎不会出错。以上种种可以看出来，通过使用学生宿舍管理系统能够更加有效的对学生宿舍信息进行管理和高效的处理，这样就可以减少了管理成本，提高了管理效率，且推动了社会信息化智能化的发展。

# 3 系统需求分析

软件需要解决的是用户所面临的现实问题，但是，这些现实问题需要由软件技术人员来解决。情况往往是，开发软件的技术人员精通计算机技术，但并不熟悉用户的业务领域；而用户清楚自己的业务，却又不太懂计算机技术。因此，对于同一个问题，技术人员和用户之间可能 存在认识上的差异。也因此，在软件技术人员着手设计软件之前，需要由既精通计算机技术又 熟悉用户应用领域的软件系统分析人员，对软件问题进行细致的需求分析。 需求分析是软件工程过程中一个重要的里程碑。在需求分析过程中，软件系统分析人员通 过研究用户在软件问题上的需求意愿，分析出软件系统在功能、性能、数据等诸多方面应该达 到的目标，从而获得有关软件的需求规格定义。

需求分析是在软件系统分析人员的操作下进行的，在这个过程中，用户和开发者之间需要达成的是对系统的一致性需求认识。实际上，可以把软件系统分析人员看成是软件用户与软件 开发技术人员之间的信息通道，其作用是使用户对软件问题的现实描述，能够有效地转变为开 发软件的技术人员所需要的对软件的技术描述，以方便技术人员对软件的技术构建。

## 3.1 功能需求分析

学生宿舍管理系统的开发采用的是JSP技术+Servlet技术+MySQL数据库管理系统。该系统利用软件开发技术的成果，方便学生宿舍相关信息的管理，根据用户的需求，快捷方便的为用户提供检索查询功能。其系统的功能主要包括：

当系统登陆时，首先出现的是一个登陆的界面，登录后的用户会有权限差别，管理员和普通用户，权限设置主要是维护系统的安全性和完整性。拥有管理员权限的用户能对其他用户进行相应的权限设置也可以维护其它用户的信息，没有权限的用户不能对相应的数据进行操作。管理员登录后，可以对各个功能模块的的信息进行管理，主要包括用户管理模块、公告管理模块、宿管员管理模块、宿舍管理模块、学生管理模块等。普通用户可以进行注册、登录、用户信息浏览、公告信息浏览、宿管员信息浏览、宿舍信息浏览、学生信息浏览等功能。

根据以上功能需求分析，通过用例图来描述系统的主要功能。构建用例模型的第一步是确定模型中的用户有哪些，确定用户的原则有：谁是系统的维护者、谁是系统的参与者等。维护者处于系统内部，对系统有绝对的控制权；而参与者一般都位于系统的外部，处于系统的控制之外。

现在确定本系统用例模型有两种，分别是管理员用户和普通用户。下面分别对这两个角色的功能进行描述：

1）管理员用户

管理员用户主要负责系统数据的管理工作。管理员用户登录后，可以对各个功能模块的的信息进行管理。例如：增加用户信息、删除用户信息、更新用户信息；增加公告信息、删除公告信息、更新公告信息；增加宿管员信息、删除宿管员信息、更新宿管员信息；增加宿舍信息、删除宿舍信息、更新宿舍信息；增加学生信息、删除学生信息、更新学生信息等

(1)用户管理：管理员可以对用户的用户名、密码、姓名、性别、手机、备注、类型等信息进行增加、删除、修改、查询列表、显示详情等管理操作；

(2)公告管理：管理员可以对公告的标题、内容、类型、创建时间等信息进行增加、删除、修改、查询列表、显示详情等管理操作；

(3)宿管员管理：管理员可以对宿管员的姓名、性别、电话、楼栋、备注等信息进行增加、删除、修改、查询列表、显示详情等管理操作；

(4)宿舍管理：管理员可以对宿舍的宿舍、楼栋、床位数、宿管员、备注等信息进行增加、删除、修改、查询列表、显示详情等管理操作；

(5)学生管理：管理员可以对学生的姓名、性别、学号、楼栋、宿舍、电话、备注等信息进行增加、删除、修改、查询列表、显示详情等管理操作

2） 普通用户

普通用户需要先在本学生宿舍管理系统上进行注册。注册后方可登录本系统，进行用户、公告、宿管员、宿舍、学生等信息的查询浏览和详情查询

## 3.2 非功能需求分析

1）系统的稳定性

系统稳定性差则预示着系统在用户使用时，可能会出现数据操作错误，页面响应时间过长或者干脆无法响应等，因此系统的稳定性是用户对系统评价的重要指标之一。该系统应采用了稳定的操作系统、数据库、运行环境等，以保证系统的稳定性。

2）系统的开放性

系统的开放性是指系统具有良好的可兼容性，可以在Windows的大部分版本的操作系统中正常的运行，另外，也支持不同浏览器版本，能够在微软的IE浏览器、谷歌浏览器等常用的浏览器上正常运行。系统的易升级和管理也是系统开放性的表现。

3）系统的实用性

系统的实用性是指系统设计开发的功能要尽量简单、实用，用户在使用系统过程中能够感受到系统的快捷、方便，不存在很多累赘和多余的操作或者功能。

# 4 相关技术介绍

本章主要对学生宿舍管理系统所涉及的相关技术进行介绍。

4.1 JSP

JSP页面由HTML代码和嵌入其中的Java代码所组成。JSP是Sun Microsystems公司研究和发布的一种动态网页开发技术，尤其适用于当前的IE浏览器等应用程序开发，方便实现分布式管理。JSP具备了Java技术的易用可靠，具有简单、可移植性的特点，所以说它是一种与平台无关的开发技术。也正是因为它适应平台广、编写简单、非常适合构造基于B/S结构的动态网页而逐渐成为因特网上的主流开发技术，已经越来越受到编程者的关注和喜爱。

4.2 JavaScript

JavaScript是一种基于对象和事件驱动并具有相对安全性的客户端脚本语言。同时也是一种广泛用于客户端Web开发的脚本语言，常用来给HTML网页添加动态功能，比如响应用户的各种操作。它最初由网景公司（Netscape）的Brendan Eich设计，是一种动态、弱类型、基于原型的语言，内置支持类。

JavaScript语言与Java语言在语法上比较相似，但随着对JavaScript的深入了解后你会发现，它们说到底是两种语言。

4.3 MySQL

MySQL是一个开放源码的小型关联式数据库管理系统，开发者为瑞典MySQL AB公司。MySQL被广泛地应用在Internet上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了MySQL作为网站数据库。MySQL数据库使用我们最为熟悉的结构化查询语言进行数据库管理。正是这些特点，MySQL数据库系统收到了商业软件用户以及广大软件开发者的青睐。

4.4 B/S模式

本系统采用的是B/S(浏览器/服务器)结构。由于需要经常对系统软件进行升级和改进，因此使用B/S架构明显有着更为便利的好处，它是随着Internet技术的兴起，对C/S结构的一种变化或者改进的结构。B/S模式使用标准TCP/IP、HTTP协议，一般只要有操作系统和浏览器，并且能连上网就行，不用再特别安装软件就可以访问本系统进行网上汽车租赁的相关操作。B/S结构是一种全新的软件系统构造技术，使得原本需要复杂专用软件才能实现的强大功能，现在只要通过浏览器技术就可以实现，便于了系统的维护，节约了开发成本。



图2-1 B/S模式应用系统网络结构图

4.5 MVC模式

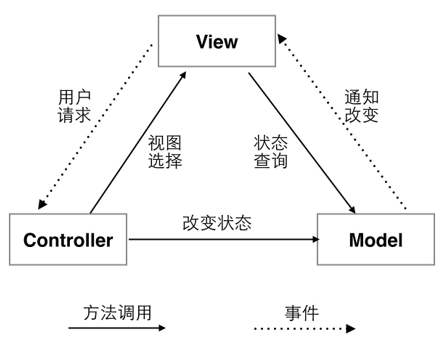
MVC即Model、View、Controller系统开发三层架构模式，有利于实现代码的高内聚低耦合的特性，也有利于后期的代码扩展与维护。

MVC的原理就是将整个系统代码的逻辑按输入、逻辑处理、输出来分为Model、View、Controller三层进行分离，从而使该系统分为视图、控制、模型层三层。

视图层指与用户交互的前端页面，例如经常使用的JSP、ThymeLeaf及FreeMarker等模板引擎。

模型层中包括逻辑上的Service(业务处理)层以及DAO(数据库交互)层，主要负责系统中业务方面以及数据交互方面的逻辑处理。

控制层负责接收接口中传来的数据以及输出数据到视图层进行页面渲染。



MVC模式应用模型

本系统部分包类功能说明：

⑴vo：即javabean，亦常命名为pojo、entity等

⑵dao/dao.impl：数据层接口及其实现，对于数据库的操作，都放到dao层，也就是dao里面通常是对数据库的增、删、改、查等操作

⑶service/service.impl：完成相应的业务逻辑处理，并调用dao层

⑷servlet：完成界面请求、对界面进行跳转等等，servlet调用service层

⑸Filter：登录过滤拦截器，对于未登录的用户，不允许其访问信息，将强制为其重定向至登录页面

⑹util：工具包，放置一些共用方法，如数据库的连接和系统常量的定义

4.6 Servlet

Java Servlet是运行在Web服务器或应用服务器上的程序，它是作为来自Web浏览器或其他HTTP客户端的请求和HTTP服务器上的数据库或应用程序之间的中间层。

使用 Servlet，我们可以收集来自网页表单的用户输入，呈现来自数据库或者其他源的记录，还可以动态创建网页。

Java Servlet 通常情况下与使用 CGI（Common Gateway Interface，公共网关接口）实现的程序可以达到异曲同工的效果。但是相比于 CGI，Servlet 有以下几点优势：

⑴性能明显更好。

⑵Servlet 在 Web 服务器的地址空间内执行。这样它就没有必要再创建一个单独的进程来处理每个客户端请求。

⑶Servlet 是独立于平台的，因为它们是用 Java 编写的。

⑷服务器上的 Java 安全管理器执行了一系列限制，以保护服务器计算机上的资源。因此，Servlet 是可信的。

⑸Java 类库的全部功能对 Servlet 来说都是可用的。它可以通过 sockets 和 RMI 机制与 applets、数据库或其他软件进行交互。

# 5 系统开发平台及运行环境

⑴IntelliJ IDEA

IDEA全称IntelliJ IDEA，是java编程语言开发的集成环境。IntelliJ在业界被公认为最好的java开发工具，尤其在智能代码助手、代码自动提示、重构、JavaEE支持、各类版本工具(git、svn等)、JUnit、CVS整合、代码分析、创新的GUI设计等方面的功能可以说是超常的。IDEA是JetBrains公司的产品，这家公司总部位于捷克共和国的首都布拉格，开发人员以严谨著称的东欧程序员为主。它的旗舰版本还支持HTML，CSS，PHP，MySQL，Python等。免费版只支持Java,Kotlin等少数语言。

⑵Tomcat服务器

Tomcat是一个小型的轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好Apache服务器，可利用它响应对HTML页面的访问请求。实际上Tomcat部分是Apache服务器的扩展，但它是独立运行的，所以当你运行Tomcat时，它实际上作为一个与Apache独立的进程单独运行的。

Tomcat服务器是一个免费的开放源代码的Web应用服务器，最新的Servlet和JSP规范总是能在Tomcat中得到体现。因为Tomcat技术先进、性能稳定，而且免费，因而深受Java爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web应用服务器。

Tomcat的环境主要有以下几方面技术优势：

a.Tomcat中的应用程序是一个WAR（WebArchive）文件。WAR是Sun提出的一种Web应用程序格式，与JAR类似，也是许多文件的一个压缩包。

b.在Tomcat中，应用程序的部署很简单，你只需将你的WAR放到Tomcat的webapp目录下，Tomcat会自动检测到这个文件，并将其解压。

c.Tomcat不仅仅是一个Servlet容器，它也具有传统的Web服务器的功能：处理html页面。

d.Tomcat也可以与其它一些软件集成起来实现更多的功能。

⑶软件方面

操作系统：Windows7+

数据库：MySQL5.5+

开发环境：JDK1.5+

开发工具：IntelliJ IDEA

应用服务器：ApacheTomcat6.0+

浏览器：IE、FireFox、Chrome

# 6 总体设计和业务流程

|  |
| --- |
| 学生宿舍管理系统 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用户管理 | 公告管理 | 宿管员管理 | 宿舍管理 | 学生管理 |

系统总体设计

失败

程序开始

注册

登录判断

登录

管理页面

成功

成功

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用户模块 | 公告模块 | 宿管员模块 | 宿舍模块 | 学生模块 | 维修信息模块 |

业务流程

# 7 数据库设计

数据库设计是要在一个给定的应用环境（DBMS）中，通过合理的逻辑设计和有效的物理设计，构造较优的数据库模式、子模式，建立数据库和设计应用程序，满足用户的各种信息需求。

数据库的设计关系到整个应用系统的运行效率，数据库设计得好，不仅有利于日常数据的维护更新，而且可以提高系统的运行效率，缩短数据查询响应周期，增加网站的流量。合理的数据库设计可以使围绕它支持的Web页面的Java代码简单化，易于实现，并且可以提高数据存储的效率，保证数据的完整一致。学生宿舍管理系统采用MySQL作为后台数据库开发工具。

## 7.1 概念设计

根据系统分析的主要实体有：用户、公告、宿管员、宿舍、学生。各个实体具体的描述属性图如下：

密码

主键

性别

用户名

备注

用户

姓名

手机

类型

用户实体图

公告

主键

内容

标题

创建时间

类型

公告实体图

宿管员

主键

姓名

性别

电话

备注

楼栋

宿管员实体图

宿舍

主键

宿舍

楼栋

床位数

备注

宿管员

宿舍实体图

性别

主键

楼栋

姓名

电话

学生

学号

宿舍

备注

学生实体图

## 7.2 表结构设计

(1) 用户表（t\_user）主要用户来维护用户相关的信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 主键 | 长度 | 说明 |
| 1 | id | bigint | 是 | 20 | 主键 |
| 2 | username | varchar | 否 | 50 | 用户名 |
| 3 | password | varchar | 否 | 50 | 密码 |
| 4 | real\_name | varchar | 否 | 50 | 姓名 |
| 5 | user\_sex | varchar | 否 | 50 | 性别 |
| 6 | user\_phone | varchar | 否 | 50 | 手机 |
| 7 | user\_text | varchar | 否 | 50 | 备注 |
| 8 | user\_type | varchar | 否 | 50 | 类型 |

(2) 公告表（t\_notice）主要用户来维护公告相关的信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 主键 | 长度 | 说明 |
| 1 | id | bigint | 是 | 20 | 主键 |
| 2 | notice\_name | varchar | 否 | 50 | 标题 |
| 3 | notice\_text | varchar | 否 | 50 | 内容 |
| 4 | notice\_type | varchar | 否 | 50 | 类型 |
| 5 | create\_date | varchar | 否 | 50 | 创建时间 |

(3) 宿管员表（t\_administrator）主要用户来维护宿管员相关的信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 主键 | 长度 | 说明 |
| 1 | id | bigint | 是 | 20 | 主键 |
| 2 | administrator\_name | varchar | 否 | 50 | 姓名 |
| 3 | administrator\_sex | varchar | 否 | 50 | 性别 |
| 4 | administrator\_phone | varchar | 否 | 50 | 电话 |
| 5 | administrator\_building | varchar | 否 | 50 | 楼栋 |
| 6 | administrator\_text | varchar | 否 | 50 | 备注 |

(4) 宿舍表（t\_dormitory）主要用户来维护宿舍相关的信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 主键 | 长度 | 说明 |
| 1 | id | bigint | 是 | 20 | 主键 |
| 2 | dormitory\_name | varchar | 否 | 50 | 宿舍 |
| 3 | dormitory\_building | varchar | 否 | 50 | 楼栋 |
| 4 | dormitory\_bedcount | varchar | 否 | 50 | 床位数 |
| 5 | dormitory\_administrator | varchar | 否 | 50 | 宿管员 |
| 6 | dormitory\_text | varchar | 否 | 50 | 备注 |

(5) 学生表（t\_student）主要用户来维护学生相关的信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 主键 | 长度 | 说明 |
| 1 | id | bigint | 是 | 20 | 主键 |
| 2 | student\_name | varchar | 否 | 50 | 姓名 |
| 3 | student\_sex | varchar | 否 | 50 | 性别 |
| 4 | student\_no | varchar | 否 | 50 | 学号 |
| 5 | student\_building | varchar | 否 | 50 | 楼栋 |
| 6 | student\_dormitory | varchar | 否 | 50 | 宿舍 |
| 7 | student\_phone | varchar | 否 | 50 | 电话 |
| 8 | student\_text | varchar | 否 | 50 | 备注 |

# 8 功能实现

## 8.1 概要设计

(1)用户管理：此模块的功能是对用户相关的信息如：用户名、密码、姓名、性别、手机、备注、类型等进行增加、删除、修改、查询列表、显示详情等管理操作，普通用户可以进行检索和查看，管理员可以对信息进行增加编辑和删除。

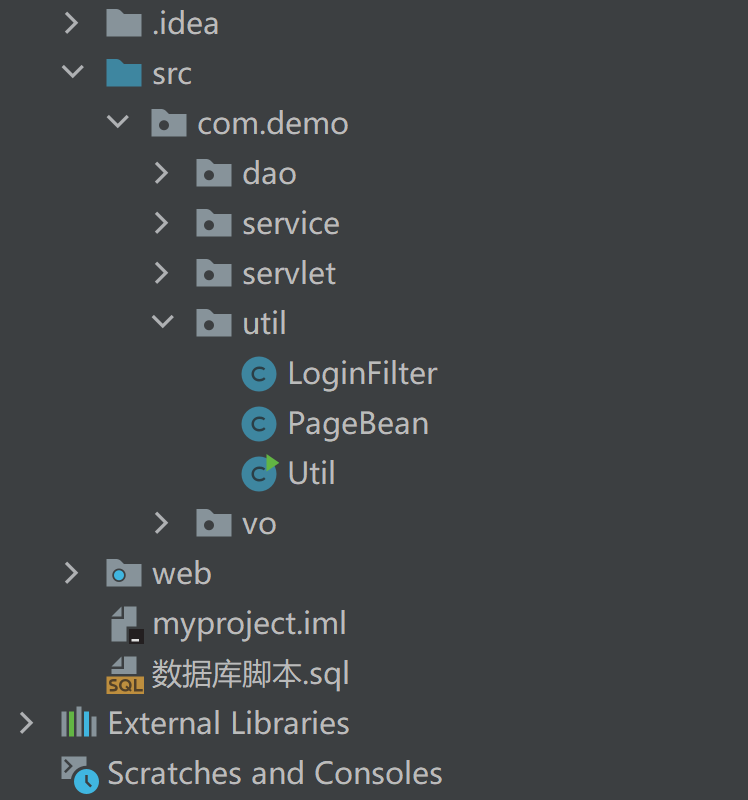
(2)公告管理：此模块的功能是对公告相关的信息如：标题、内容、类型、创建时间等进行增加、删除、修改、查询列表、显示详情等管理操作，普通用户可以进行检索和查看，管理员可以对信息进行增加编辑和删除。

(3)宿管员管理：此模块的功能是对宿管员相关的信息如：姓名、性别、电话、楼栋、备注等进行增加、删除、修改、查询列表、显示详情等管理操作，普通用户可以进行检索和查看，管理员可以对信息进行增加编辑和删除。

(4)宿舍管理：此模块的功能是对宿舍相关的信息如：宿舍、楼栋、床位数、宿管员、备注等进行增加、删除、修改、查询列表、显示详情等管理操作，普通用户可以进行检索和查看，管理员可以对信息进行增加编辑和删除。

(5)学生管理：此模块的功能是对学生相关的信息如：姓名、性别、学号、楼栋、宿舍、电话、备注等进行增加、删除、修改、查询列表、显示详情等管理操作，普通用户可以进行检索和查看，管理员可以对信息进行增加编辑和删除。

## 8.2 项目截图



## 8.3 代码实现

### 8.3.1 用户模块

数据层接口（UserDAO.java）

|  |
| --- |
| package com.demo.dao;  import com.demo.vo.User;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* User模块的DAO层（数据层）接口，提供增删改查等数据库操作的方法抽象  \*/  public interface UserDAO {  /\*\*  \* 增加用户表记录  \*  \* @param vo  \* @return  \*/  void add(User vo);  /\*\*  \* 根据主键id，删除对应的用户表记录  \*  \* @param id  \* @return  \*/  boolean delete(long id);  /\*\*  \* 更新用户表记录  \*  \* @param vo  \* @return  \*/  void update(User vo);  /\*\*  \* 根据主键id获取用户表记录的详情  \*  \* @param id  \* @return  \*/  User get(Serializable id);  /\*\*  \* 根据条件查询用户的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

数据层实现类（UserDAOImpl.java）

|  |
| --- |
| package com.demo.dao.impl;  import com.demo.util.Util;  import com.demo.dao.UserDAO;  import com.demo.vo.User;  import java.io.Serializable;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* User模块的DAO层（数据层）的具体实现类，对UserDAO接口中定义的增删改查等抽象方法作出具体的功能实现  \*/  public class UserDAOImpl implements UserDAO {  //@Override  public void add(User vo) {  String sql = "insert into `t\_user` (`username`,`password`,`real\_name`,`user\_sex`,`user\_phone`,`user\_text`,`user\_type`) values(?,?,?,?,?,?,?)";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getUsername());  ps.setString(2, vo.getPassword());  ps.setString(3, vo.getRealName());  ps.setString(4, vo.getUserSex());  ps.setString(5, vo.getUserPhone());  ps.setString(6, vo.getUserText());  ps.setString(7, vo.getUserType());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public void update(User vo) {  String sql = "update `t\_user` set `username` = ? ,`password` = ? ,`real\_name` = ? ,`user\_sex` = ? ,`user\_phone` = ? ,`user\_text` = ? ,`user\_type` = ? where `id` = ?";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getUsername());  ps.setString(2, vo.getPassword());  ps.setString(3, vo.getRealName());  ps.setString(4, vo.getUserSex());  ps.setString(5, vo.getUserPhone());  ps.setString(6, vo.getUserText());  ps.setString(7, vo.getUserType());  ps.setLong(8, vo.getId());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public boolean delete(long id) {  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "delete from `t\_user` where id = " + id;  s.execute(sql);  s.close();  c.close();  return true;  } catch (Exception e) {  e.printStackTrace();  return false;  }  }  //@Override  public User get(Serializable id) {  User vo = null;  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "select \* from `t\_user` where id = " + id;  ResultSet rs = s.executeQuery(sql);  if (rs.next()) {  vo = new User();  vo.setId(rs.getLong("id"));  vo.setUsername(rs.getString("username"));  vo.setPassword(rs.getString("password"));  vo.setRealName(rs.getString("real\_name"));  vo.setUserSex(rs.getString("user\_sex"));  vo.setUserPhone(rs.getString("user\_phone"));  vo.setUserText(rs.getString("user\_text"));  vo.setUserType(rs.getString("user\_type"));  }  c.close();  } catch (Exception e) {  e.printStackTrace();  }  return vo;  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  List<User> list = new ArrayList();  int totalCount = 0;  String condition = "";  String sqlList;  if (params.get("searchColumn") != null && !"".equals(params.get("searchColumn"))) {  condition += " and `" + params.get("searchColumn") + "` like '%" + params.get("keyword") + "%'";  }  try {  Connection c = Util.getConnection();  PreparedStatement ps;  ResultSet rs;  String limit = (params.get("startIndex") != null && params.get("pageSize") != null) ? " limit " + params.get("startIndex") + "," + params.get("pageSize") : "";  sqlList = "select \* from `t\_user` where 1=1 " + condition + " order by id asc " + limit + ";";  ps = c.prepareStatement(sqlList);  rs = ps.executeQuery();  while (rs.next()) {  User vo = new User();  vo.setId(rs.getLong("id"));  vo.setUsername(rs.getString("username"));  vo.setPassword(rs.getString("password"));  vo.setRealName(rs.getString("real\_name"));  vo.setUserSex(rs.getString("user\_sex"));  vo.setUserPhone(rs.getString("user\_phone"));  vo.setUserText(rs.getString("user\_text"));  vo.setUserType(rs.getString("user\_type"));  list.add(vo);  }  String sqlCount = "select count(\*) from `t\_user` where 1=1 " + condition;  ps = c.prepareStatement(sqlCount);  rs = ps.executeQuery();  if (rs.next()) {  totalCount = rs.getInt(1);  }  rs.close();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  Map<String, Object> result = new HashMap();  result.put("list", list);  result.put("totalCount", totalCount);  return result;  }  } |

业务层接口（UserService.java）

|  |
| --- |
| package com.demo.service;  import com.demo.vo.User;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* User模块的Service层（业务层）接口，提供业务方法的抽象  \*/  public interface UserService {  /\*\*  \* 增加用户  \*  \* @param vo  \* @return  \*/  void add(User vo);  /\*\*  \* 删除用户  \*  \* @param id  \* @return  \*/  void delete(long id);  /\*\*  \* 修改用户  \*  \* @param vo  \* @return  \*/  void update(User vo);  /\*\*  \* 根据主键Id查询用户详情  \*  \* @param id  \* @return  \*/  User get(Serializable id);  /\*\*  \* 根据条件查询用户的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

业务层实现类（UserServiceImpl.java）

|  |
| --- |
| package com.demo.service.impl;  import com.demo.dao.UserDAO;  import com.demo.dao.impl.UserDAOImpl;  import com.demo.service.UserService;  import com.demo.vo.User;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* User模块的Service层（业务层）的具体实现类，对UserService接口中定义的抽象方法作出具体的功能实现  \*/  public class UserServiceImpl implements UserService {  //@Override  public void add(User vo) {  UserDAO userDAO = new UserDAOImpl();  userDAO.add(vo);  }  //@Override  public void delete(long id) {  UserDAO userDAO = new UserDAOImpl();  userDAO.delete(id);  }  //@Override  public void update(User vo) {  UserDAO userDAO = new UserDAOImpl();  userDAO.update(vo);  }  //@Override  public User get(Serializable id) {  UserDAO userDAO = new UserDAOImpl();  return userDAO.get(id);  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  UserDAO userDAO = new UserDAOImpl();  return userDAO.list(params);  }  } |

控制层（UserServlet.java）

|  |
| --- |
| package com.demo.servlet;  import com.demo.util.Util;  import com.demo.service.UserService;  import com.demo.service.impl.UserServiceImpl;  import com.demo.vo.User;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  import java.io.IOException;  import java.io.Serializable;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* User模块的Servlet控制层，负责接收页面传过来的请求参数，根据action参数的值来确定页面要执行的具体操作<br>  \* 而后再调用UserService业务层的方法来处理具体的业务，最后将处理完成的结果返回或跳转至相应页面  \*/  //@WebServlet("/UserServlet")  public class UserServlet extends HttpServlet {  /\*\*  \* 处理Post请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //过滤编码  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  String action = Util.decode(request, "action");  if ("add".equals(action)) {//增加  User vo = new User();  //取出页面传进来的各个数据，并设置到User对象的属性里  String username = Util.decode(request, "username");  if (username != null && !username.equals("")) {  vo.setUsername(username);  }  String password = Util.decode(request, "password");  if (password != null && !password.equals("")) {  vo.setPassword(password);  }  String realName = Util.decode(request, "realName");  if (realName != null && !realName.equals("")) {  vo.setRealName(realName);  }  String userSex = Util.decode(request, "userSex");  if (userSex != null && !userSex.equals("")) {  vo.setUserSex(userSex);  }  String userPhone = Util.decode(request, "userPhone");  if (userPhone != null && !userPhone.equals("")) {  vo.setUserPhone(userPhone);  }  String userText = Util.decode(request, "userText");  if (userText != null && !userText.equals("")) {  vo.setUserText(userText);  }  String userType = Util.decode(request, "userType");  if (userType != null && !userType.equals("")) {  vo.setUserType(userType);  }  UserService userService = new UserServiceImpl();  //调用Service层增加方法（add），增加记录  userService.add(vo);  this.redirectList(request, response);  } else if ("delete".equals(action)) {//删除  //取出表要删除的用户记录的主键  long id = Long.parseLong(Util.decode(request, "id"));  UserService userService = new UserServiceImpl();  //调用Service层删除方法（delete），将对应的记录删除  userService.delete(id);  this.redirectList(request, response);  } else if ("edit".equals(action)) {//修改  //取出页面传进来的各个数据，并设置到User对象的属性里  User vo = new User();  String id = Util.decode(request, "id");  if (id != null && !id.equals("")) {  vo.setId(Long.valueOf(id));  }  String username = Util.decode(request, "username");  if (username != null && !username.equals("")) {  vo.setUsername(username);  }  String password = Util.decode(request, "password");  if (password != null && !password.equals("")) {  vo.setPassword(password);  }  String realName = Util.decode(request, "realName");  if (realName != null && !realName.equals("")) {  vo.setRealName(realName);  }  String userSex = Util.decode(request, "userSex");  if (userSex != null && !userSex.equals("")) {  vo.setUserSex(userSex);  }  String userPhone = Util.decode(request, "userPhone");  if (userPhone != null && !userPhone.equals("")) {  vo.setUserPhone(userPhone);  }  String userText = Util.decode(request, "userText");  if (userText != null && !userText.equals("")) {  vo.setUserText(userText);  }  String userType = Util.decode(request, "userType");  if (userType != null && !userType.equals("")) {  vo.setUserType(userType);  }  UserService userService = new UserServiceImpl();  //调用Service层更新方法（update），更新记录  userService.update(vo);  this.redirectList(request, response);  } else if ("get".equalsIgnoreCase(action) || "editPre".equalsIgnoreCase(action)) {//根据主键ID，查询详情信息并跳转到详情页面或编辑页面  Serializable id = Util.decode(request, "id");//取出页面传入的主键，用于查询详情  UserService userService = new UserServiceImpl();  User vo = userService.get(id);  request.getSession().setAttribute("vo", vo);  String to = "get".equalsIgnoreCase(action) ? "info" : "edit";//判断是去详情显示页面还是编辑页面  response.sendRedirect("user\_" + to + ".jsp");  } else {//默认去列表页面  this.redirectList(request, response);  }  }  /\*\*  \* 处理Get请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doPost(request, response);//Get请求和Post请求的处理是一样的，所以把request、response转交给Post方法就好  }  /\*\*  \* 根据参数，查询出条例条件的记录集合，最后将数据返回给调用处或者将数据集合设置到session域里，再跳转到对应的列表页面  \*  \* @param request  \* @param response  \*/  private void redirectList(HttpServletRequest request, HttpServletResponse response) throws IOException {  //查询列和关键字  String searchColumn = Util.decode(request, "searchColumn");  String keyword = Util.decode(request, "keyword");  Map<String, Object> params = new HashMap();//用来保存控制层传进来的参数(查询条件)  params.put("searchColumn", searchColumn);//要查询的列  params.put("keyword", keyword);//查询的关键字  UserService userService = new UserServiceImpl();  Map<String, Object> map = userService.list(params);  request.getSession().setAttribute("list", map.get("list"));  Integer totalRecord = (Integer) map.get("totalCount");//根据查询条件取出对应的总记录数，用于分页  String pageNum = Util.decode(request, "pageNum");//封装分页参数  com.demo.util.PageBean<Object> pb = new com.demo.util.PageBean(Integer.valueOf(pageNum != null ? pageNum : "1"), totalRecord);  params.put("startIndex", pb.getStartIndex());  params.put("pageSize", pb.getPageSize());  List list = (List) userService.list(params).get("list");//根据分页参数startIndex、pageSize查询出来的最终结果list  pb.setServlet("UserServlet");  pb.setSearchColumn(searchColumn);  pb.setKeyword(keyword);  pb.setList(list);  request.getSession().setAttribute("pageBean", pb);  request.getSession().setAttribute("list", pb.getList());  response.sendRedirect("user\_list.jsp");  }  } |

JavaBean实体模型层定义（User.java）

|  |
| --- |
| package com.demo.vo;  import java.io.Serializable;  /\*\*  \* 用户（t\_user表对应的Java实体类）  \*/  public class User implements Serializable {  private Long id;//主键  private String username;//用户名  private String password;//密码  private String realName;//姓名  private String userSex;//性别:男/女  private String userPhone;//手机  private String userText;//备注  private String userType;//类型:管理员/普通用户  public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getRealName() {  return realName;  }  public void setRealName(String realName) {  this.realName = realName;  }  public String getUserSex() {  return userSex;  }  public void setUserSex(String userSex) {  this.userSex = userSex;  }  public String getUserPhone() {  return userPhone;  }  public void setUserPhone(String userPhone) {  this.userPhone = userPhone;  }  public String getUserText() {  return userText;  }  public void setUserText(String userText) {  this.userText = userText;  }  public String getUserType() {  return userType;  }  public void setUserType(String userType) {  this.userType = userType;  }  } |

增加信息页面（user\_add.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>添加用户</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">添加用户</a>  <br>  <br>  </div>  <br>  <form action="UserServlet?action=add" method="post" onsubmit="return check()">  <table class="index-content-table-add">  <tr>  <td width="12%">用户名：</td><td><input class="index-content-table-td-add" type="text" id="username" name="username" value=""/></td>  </tr>  <tr>  <td width="12%">密码：</td><td><input class="index-content-table-td-add" type="text" id="password" name="password" value=""/></td>  </tr>  <tr>  <td width="12%">姓名：</td><td><input class="index-content-table-td-add" type="text" id="realName" name="realName" value=""/></td>  </tr>  <tr>  <td width="12%">性别：</td>  <td>  <input name="userSex" type="radio" value="男" checked="checked"/>&nbsp;&nbsp;&nbsp;男&nbsp;&nbsp;&nbsp;&nbsp;  <input name="userSex" type="radio" value="女"/>&nbsp;&nbsp;&nbsp;女&nbsp;&nbsp;&nbsp;&nbsp;  </td>  </tr>  <tr>  <td width="12%">手机：</td><td><input class="index-content-table-td-add" type="text" id="userPhone" name="userPhone" value=""/></td>  </tr>  <tr>  <td width="12%">备注：</td><td><textarea id="userText" name="userText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......"></textarea></td>  </tr>  <tr>  <td width="12%">类型：</td>  <td>  <input name="userType" type="radio" value="管理员" checked="checked"/>&nbsp;&nbsp;&nbsp;管理员&nbsp;&nbsp;&nbsp;&nbsp;  <input name="userType" type="radio" value="普通用户"/>&nbsp;&nbsp;&nbsp;普通用户&nbsp;&nbsp;&nbsp;&nbsp;  </td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("username").value.trim().length == 0) {  alert("用户名不能为空!");  return false;  }  if (document.getElementById("password").value.trim().length == 0) {  alert("密码不能为空!");  return false;  }  if (document.getElementById("realName").value.trim().length == 0) {  alert("姓名不能为空!");  return false;  }  if (document.getElementById("userPhone").value.trim().length == 0) {  alert("手机不能为空!");  return false;  }  return true;  }  </script>  </html> |

更新信息页面（user\_edit.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>修改用户</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">修改用户</a>  <br>  <br>  </div>  <br>  <form action="UserServlet?action=edit" method="post" onsubmit="return check()">  <input type="hidden" id="id" name="id" value="${vo.id}"/>    <table class="index-content-table-add">  <tr>  <td width="12%">用户名：</td><td><input class="index-content-table-td-add" type="text" id="username" name="username" value="${vo.username}"/></td>  </tr>  <tr>  <td width="12%">密码：</td><td><input class="index-content-table-td-add" type="text" id="password" name="password" value="${vo.password}"/></td>  </tr>  <tr>  <td width="12%">姓名：</td><td><input class="index-content-table-td-add" type="text" id="realName" name="realName" value="${vo.realName}"/></td>  </tr>  <tr>  <td width="12%">性别：</td>  <td>  <input name="userSex" type="radio" value="男" ${vo.userSex=='男'?'checked':''}/>&nbsp;&nbsp;&nbsp;男&nbsp;&nbsp;&nbsp;&nbsp;  <input name="userSex" type="radio" value="女" ${vo.userSex=='女'?'checked':''}/>&nbsp;&nbsp;&nbsp;女&nbsp;&nbsp;&nbsp;&nbsp;  </td>  </tr>  <tr>  <td width="12%">手机：</td><td><input class="index-content-table-td-add" type="text" id="userPhone" name="userPhone" value="${vo.userPhone}"/></td>  </tr>  <tr>  <td width="12%">备注：</td><td><textarea id="userText" name="userText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......">${vo.userText}</textarea></td>  </tr>  <tr>  <td width="12%">类型：</td>  <td>  <input <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> name="userType" type="radio" value="管理员" ${vo.userType=='管理员'?'checked':''}/>&nbsp;&nbsp;&nbsp;管理员&nbsp;&nbsp;&nbsp;&nbsp;  <input name="userType" type="radio" value="普通用户" ${vo.userType=='普通用户'?'checked':''}/>&nbsp;&nbsp;&nbsp;普通用户&nbsp;&nbsp;&nbsp;&nbsp;  </td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("username").value.trim().length == 0) {  alert("用户名不能为空!");  return false;  }  if (document.getElementById("password").value.trim().length == 0) {  alert("密码不能为空!");  return false;  }  if (document.getElementById("realName").value.trim().length == 0) {  alert("姓名不能为空!");  return false;  }  if (document.getElementById("userPhone").value.trim().length == 0) {  alert("手机不能为空!");  return false;  }  return true;  }  </script>  </html> |

查看详情页面（user\_info.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>用户详情</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">用户详情</a>  <br>  <br>  </div>  <br>  <form>  <table class="index-content-table-add" style="font-size: 18px;">  <tr>  <td>用户名：<b>${vo.username}</b></td>  </tr>  <tr>  <td>姓名：<b>${vo.realName}</b></td>  </tr>  <tr>  <td>性别：  <b>${vo.userSex}</b>  </td>  </tr>  <tr>  <td>手机：<b>${vo.userPhone}</b></td>  </tr>  <tr>  <td>备注：<b>${vo.userText}</b></td>  </tr>  <tr>  <td>类型：  <b>${vo.userType}</b>  </td>  </tr>  </table>  <br>  <button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">返回</button>  </form>  </div>  </body>  </html> |

列表与查询页面（user\_list.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>用户管理</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">用户管理</a>  <br>  <br>  </div>  <br>  <div class="index-content-operation">  <button class="btn btn-default btn-sm" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="window.location.href='user\_add.jsp'">添加</button>  <div class="index-content-operation-search"><input id="search\_keyword" placeholder="姓名" type="text" name="search\_keyword"/><input type="hidden" id="searchColumn" name="searchColumn" value="real\_name"/><button class="btn btn-primary btn-sm" onclick="searchList()">搜索</button></div>  </div>  <br>  <table class="table table-striped table-hover table-bordered">  <thead>  <tr class="index-content-table-th">  <th>用户名</th>  <th>姓名</th>  <th>性别</th>  <th>手机</th>  <th>类型</th>  <th>操作</th>  </tr>  </thead>  <tbody>  <c:forEach items="${list}" var="vo">  <tr class="index-content-table-td">  <td>${vo.username}</td>  <td>${vo.realName}</td>  <td>${vo.userSex}</td>  <td>${vo.userPhone}</td>  <td>${vo.userType}</td>  <td>  <button class="btn btn-success btn-sm" style="padding: 0px 1px;" onclick="window.location.href='UserServlet?action=get&id=${vo.id}'">详情</button>&nbsp;  <button class="btn btn-info btn-sm" style="padding: 0px 1px;"  <c:if test="${loginUser.userType != '管理员' && vo.id != loginUser.id}">disabled="disabled" title="没有权限！！！"</c:if>  onclick="window.location.href='UserServlet?action=editPre&id=${vo.id}'">编辑</button>&nbsp;  <button class="btn btn-danger btn-sm" style="padding: 0px 1px;" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="if(window.confirm('将要删除：${vo.realName}？'))window.location.href='UserServlet?action=delete&id=${vo.id}'">删除</button>  </td>  </tr>  </c:forEach>  </tbody>  </table>  <div style="float: right;padding-right: 10px;color: #515151;"><jsp:include page="split.jsp"/></div>  </div>  </body>  <script>  function searchList() {  window.location.href = "UserServlet?action=list&searchColumn="+document.getElementById("searchColumn").value+"&keyword=" + document.getElementById("search\_keyword").value;  }  </script>  </html> |

### 8.3.2 公告模块

数据层接口（NoticeDAO.java）

|  |
| --- |
| package com.demo.dao;  import com.demo.vo.Notice;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Notice模块的DAO层（数据层）接口，提供增删改查等数据库操作的方法抽象  \*/  public interface NoticeDAO {  /\*\*  \* 增加公告表记录  \*  \* @param vo  \* @return  \*/  void add(Notice vo);  /\*\*  \* 根据主键id，删除对应的公告表记录  \*  \* @param id  \* @return  \*/  boolean delete(long id);  /\*\*  \* 更新公告表记录  \*  \* @param vo  \* @return  \*/  void update(Notice vo);  /\*\*  \* 根据主键id获取公告表记录的详情  \*  \* @param id  \* @return  \*/  Notice get(Serializable id);  /\*\*  \* 根据条件查询公告的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

数据层实现类（NoticeDAOImpl.java）

|  |
| --- |
| package com.demo.dao.impl;  import com.demo.util.Util;  import com.demo.dao.NoticeDAO;  import com.demo.vo.Notice;  import java.io.Serializable;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* Notice模块的DAO层（数据层）的具体实现类，对NoticeDAO接口中定义的增删改查等抽象方法作出具体的功能实现  \*/  public class NoticeDAOImpl implements NoticeDAO {  //@Override  public void add(Notice vo) {  String sql = "insert into `t\_notice` (`notice\_name`,`notice\_text`,`notice\_type`,`create\_date`) values(?,?,?,?)";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getNoticeName());  ps.setString(2, vo.getNoticeText());  ps.setString(3, vo.getNoticeType());  ps.setString(4, vo.getCreateDate());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public void update(Notice vo) {  String sql = "update `t\_notice` set `notice\_name` = ? ,`notice\_text` = ? ,`notice\_type` = ? ,`create\_date` = ? where `id` = ?";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getNoticeName());  ps.setString(2, vo.getNoticeText());  ps.setString(3, vo.getNoticeType());  ps.setString(4, vo.getCreateDate());  ps.setLong(5, vo.getId());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public boolean delete(long id) {  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "delete from `t\_notice` where id = " + id;  s.execute(sql);  s.close();  c.close();  return true;  } catch (Exception e) {  e.printStackTrace();  return false;  }  }  //@Override  public Notice get(Serializable id) {  Notice vo = null;  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "select \* from `t\_notice` where id = " + id;  ResultSet rs = s.executeQuery(sql);  if (rs.next()) {  vo = new Notice();  vo.setId(rs.getLong("id"));  vo.setNoticeName(rs.getString("notice\_name"));  vo.setNoticeText(rs.getString("notice\_text"));  vo.setNoticeType(rs.getString("notice\_type"));  vo.setCreateDate(rs.getString("create\_date"));  }  c.close();  } catch (Exception e) {  e.printStackTrace();  }  return vo;  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  List<Notice> list = new ArrayList();  int totalCount = 0;  String condition = "";  String sqlList;  if (params.get("searchColumn") != null && !"".equals(params.get("searchColumn"))) {  condition += " and `" + params.get("searchColumn") + "` like '%" + params.get("keyword") + "%'";  }  try {  Connection c = Util.getConnection();  PreparedStatement ps;  ResultSet rs;  String limit = (params.get("startIndex") != null && params.get("pageSize") != null) ? " limit " + params.get("startIndex") + "," + params.get("pageSize") : "";  sqlList = "select \* from `t\_notice` where 1=1 " + condition + " order by id asc " + limit + ";";  ps = c.prepareStatement(sqlList);  rs = ps.executeQuery();  while (rs.next()) {  Notice vo = new Notice();  vo.setId(rs.getLong("id"));  vo.setNoticeName(rs.getString("notice\_name"));  vo.setNoticeText(rs.getString("notice\_text"));  vo.setNoticeType(rs.getString("notice\_type"));  vo.setCreateDate(rs.getString("create\_date"));  list.add(vo);  }  String sqlCount = "select count(\*) from `t\_notice` where 1=1 " + condition;  ps = c.prepareStatement(sqlCount);  rs = ps.executeQuery();  if (rs.next()) {  totalCount = rs.getInt(1);  }  rs.close();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  Map<String, Object> result = new HashMap();  result.put("list", list);  result.put("totalCount", totalCount);  return result;  }  } |

业务层接口（NoticeService.java）

|  |
| --- |
| package com.demo.service;  import com.demo.vo.Notice;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Notice模块的Service层（业务层）接口，提供业务方法的抽象  \*/  public interface NoticeService {  /\*\*  \* 增加公告  \*  \* @param vo  \* @return  \*/  void add(Notice vo);  /\*\*  \* 删除公告  \*  \* @param id  \* @return  \*/  void delete(long id);  /\*\*  \* 修改公告  \*  \* @param vo  \* @return  \*/  void update(Notice vo);  /\*\*  \* 根据主键Id查询公告详情  \*  \* @param id  \* @return  \*/  Notice get(Serializable id);  /\*\*  \* 根据条件查询公告的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

业务层实现类（NoticeServiceImpl.java）

|  |
| --- |
| package com.demo.service.impl;  import com.demo.dao.NoticeDAO;  import com.demo.dao.impl.NoticeDAOImpl;  import com.demo.service.NoticeService;  import com.demo.vo.Notice;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Notice模块的Service层（业务层）的具体实现类，对NoticeService接口中定义的抽象方法作出具体的功能实现  \*/  public class NoticeServiceImpl implements NoticeService {  //@Override  public void add(Notice vo) {  NoticeDAO noticeDAO = new NoticeDAOImpl();  noticeDAO.add(vo);  }  //@Override  public void delete(long id) {  NoticeDAO noticeDAO = new NoticeDAOImpl();  noticeDAO.delete(id);  }  //@Override  public void update(Notice vo) {  NoticeDAO noticeDAO = new NoticeDAOImpl();  noticeDAO.update(vo);  }  //@Override  public Notice get(Serializable id) {  NoticeDAO noticeDAO = new NoticeDAOImpl();  return noticeDAO.get(id);  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  NoticeDAO noticeDAO = new NoticeDAOImpl();  return noticeDAO.list(params);  }  } |

控制层（NoticeServlet.java）

|  |
| --- |
| package com.demo.servlet;  import com.demo.util.Util;  import com.demo.service.NoticeService;  import com.demo.service.impl.NoticeServiceImpl;  import com.demo.vo.Notice;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  import java.io.IOException;  import java.io.Serializable;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* Notice模块的Servlet控制层，负责接收页面传过来的请求参数，根据action参数的值来确定页面要执行的具体操作<br>  \* 而后再调用NoticeService业务层的方法来处理具体的业务，最后将处理完成的结果返回或跳转至相应页面  \*/  //@WebServlet("/NoticeServlet")  public class NoticeServlet extends HttpServlet {  /\*\*  \* 处理Post请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //过滤编码  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  String action = Util.decode(request, "action");  if ("add".equals(action)) {//增加  Notice vo = new Notice();  //取出页面传进来的各个数据，并设置到Notice对象的属性里  String noticeName = Util.decode(request, "noticeName");  if (noticeName != null && !noticeName.equals("")) {  vo.setNoticeName(noticeName);  }  String noticeText = Util.decode(request, "noticeText");  if (noticeText != null && !noticeText.equals("")) {  vo.setNoticeText(noticeText);  }  String noticeType = Util.decode(request, "noticeType");  if (noticeType != null && !noticeType.equals("")) {  vo.setNoticeType(noticeType);  }  String createDate = Util.decode(request, "createDate");  if (createDate != null && !createDate.equals("")) {  vo.setCreateDate(createDate);  }  NoticeService noticeService = new NoticeServiceImpl();  //调用Service层增加方法（add），增加记录  noticeService.add(vo);  this.redirectList(request, response);  } else if ("delete".equals(action)) {//删除  //取出表要删除的公告记录的主键  long id = Long.parseLong(Util.decode(request, "id"));  NoticeService noticeService = new NoticeServiceImpl();  //调用Service层删除方法（delete），将对应的记录删除  noticeService.delete(id);  this.redirectList(request, response);  } else if ("edit".equals(action)) {//修改  //取出页面传进来的各个数据，并设置到Notice对象的属性里  Notice vo = new Notice();  String id = Util.decode(request, "id");  if (id != null && !id.equals("")) {  vo.setId(Long.valueOf(id));  }  String noticeName = Util.decode(request, "noticeName");  if (noticeName != null && !noticeName.equals("")) {  vo.setNoticeName(noticeName);  }  String noticeText = Util.decode(request, "noticeText");  if (noticeText != null && !noticeText.equals("")) {  vo.setNoticeText(noticeText);  }  String noticeType = Util.decode(request, "noticeType");  if (noticeType != null && !noticeType.equals("")) {  vo.setNoticeType(noticeType);  }  String createDate = Util.decode(request, "createDate");  if (createDate != null && !createDate.equals("")) {  vo.setCreateDate(createDate);  }  NoticeService noticeService = new NoticeServiceImpl();  //调用Service层更新方法（update），更新记录  noticeService.update(vo);  this.redirectList(request, response);  } else if ("get".equalsIgnoreCase(action) || "editPre".equalsIgnoreCase(action)) {//根据主键ID，查询详情信息并跳转到详情页面或编辑页面  Serializable id = Util.decode(request, "id");//取出页面传入的主键，用于查询详情  NoticeService noticeService = new NoticeServiceImpl();  Notice vo = noticeService.get(id);  request.getSession().setAttribute("vo", vo);  String to = "get".equalsIgnoreCase(action) ? "info" : "edit";//判断是去详情显示页面还是编辑页面  response.sendRedirect("notice\_" + to + ".jsp");  } else {//默认去列表页面  this.redirectList(request, response);  }  }  /\*\*  \* 处理Get请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doPost(request, response);//Get请求和Post请求的处理是一样的，所以把request、response转交给Post方法就好  }  /\*\*  \* 根据参数，查询出条例条件的记录集合，最后将数据返回给调用处或者将数据集合设置到session域里，再跳转到对应的列表页面  \*  \* @param request  \* @param response  \*/  private void redirectList(HttpServletRequest request, HttpServletResponse response) throws IOException {  //查询列和关键字  String searchColumn = Util.decode(request, "searchColumn");  String keyword = Util.decode(request, "keyword");  Map<String, Object> params = new HashMap();//用来保存控制层传进来的参数(查询条件)  params.put("searchColumn", searchColumn);//要查询的列  params.put("keyword", keyword);//查询的关键字  NoticeService noticeService = new NoticeServiceImpl();  Map<String, Object> map = noticeService.list(params);  request.getSession().setAttribute("list", map.get("list"));  Integer totalRecord = (Integer) map.get("totalCount");//根据查询条件取出对应的总记录数，用于分页  String pageNum = Util.decode(request, "pageNum");//封装分页参数  com.demo.util.PageBean<Object> pb = new com.demo.util.PageBean(Integer.valueOf(pageNum != null ? pageNum : "1"), totalRecord);  params.put("startIndex", pb.getStartIndex());  params.put("pageSize", pb.getPageSize());  List list = (List) noticeService.list(params).get("list");//根据分页参数startIndex、pageSize查询出来的最终结果list  pb.setServlet("NoticeServlet");  pb.setSearchColumn(searchColumn);  pb.setKeyword(keyword);  pb.setList(list);  request.getSession().setAttribute("pageBean", pb);  request.getSession().setAttribute("list", pb.getList());  response.sendRedirect("notice\_list.jsp");  }  } |

JavaBean实体模型层定义（Notice.java）

|  |
| --- |
| package com.demo.vo;  import java.io.Serializable;  /\*\*  \* 公告（t\_notice表对应的Java实体类）  \*/  public class Notice implements Serializable {  private Long id;//主键  private String noticeName;//标题  private String noticeText;//内容  private String noticeType;//类型  private String createDate;//创建时间  public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public String getNoticeName() {  return noticeName;  }  public void setNoticeName(String noticeName) {  this.noticeName = noticeName;  }  public String getNoticeText() {  return noticeText;  }  public void setNoticeText(String noticeText) {  this.noticeText = noticeText;  }  public String getNoticeType() {  return noticeType;  }  public void setNoticeType(String noticeType) {  this.noticeType = noticeType;  }  public String getCreateDate() {  return createDate;  }  public void setCreateDate(String createDate) {  this.createDate = createDate;  }  } |

增加信息页面（notice\_add.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>添加公告</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">添加公告</a>  <br>  <br>  </div>  <br>  <form action="NoticeServlet?action=add" method="post" onsubmit="return check()">  <table class="index-content-table-add">  <tr>  <td width="12%">标题：</td><td><input class="index-content-table-td-add" type="text" id="noticeName" name="noticeName" value=""/></td>  </tr>  <tr>  <td width="12%">内容：</td><td><textarea id="noticeText" name="noticeText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......"></textarea></td>  </tr>  <tr>  <td width="12%">类型：</td><td><input class="index-content-table-td-add" type="text" id="noticeType" name="noticeType" value=""/></td>  </tr>  <tr>  <td width="12%">创建时间：</td><td><input class="index-content-table-td-add" type="text" id="createDate" name="createDate" value=""/></td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("noticeName").value.trim().length == 0) {  alert("标题不能为空!");  return false;  }  if (document.getElementById("noticeType").value.trim().length == 0) {  alert("类型不能为空!");  return false;  }  if (document.getElementById("createDate").value.trim().length == 0) {  alert("创建时间不能为空!");  return false;  }  return true;  }  </script>  </html> |

更新信息页面（notice\_edit.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>修改公告</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">修改公告</a>  <br>  <br>  </div>  <br>  <form action="NoticeServlet?action=edit" method="post" onsubmit="return check()">  <input type="hidden" id="id" name="id" value="${vo.id}"/>    <table class="index-content-table-add">  <tr>  <td width="12%">标题：</td><td><input class="index-content-table-td-add" type="text" id="noticeName" name="noticeName" value="${vo.noticeName}"/></td>  </tr>  <tr>  <td width="12%">内容：</td><td><textarea id="noticeText" name="noticeText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......">${vo.noticeText}</textarea></td>  </tr>  <tr>  <td width="12%">类型：</td><td><input class="index-content-table-td-add" type="text" id="noticeType" name="noticeType" value="${vo.noticeType}"/></td>  </tr>  <tr>  <td width="12%">创建时间：</td><td><input class="index-content-table-td-add" type="text" id="createDate" name="createDate" value="${vo.createDate}"/></td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("noticeName").value.trim().length == 0) {  alert("标题不能为空!");  return false;  }  if (document.getElementById("noticeType").value.trim().length == 0) {  alert("类型不能为空!");  return false;  }  if (document.getElementById("createDate").value.trim().length == 0) {  alert("创建时间不能为空!");  return false;  }  return true;  }  </script>  </html> |

查看详情页面（notice\_info.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>公告详情</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">公告详情</a>  <br>  <br>  </div>  <br>  <form>  <table class="index-content-table-add" style="font-size: 18px;">  <tr>  <td>标题：<b>${vo.noticeName}</b></td>  </tr>  <tr>  <td>内容：<b>${vo.noticeText}</b></td>  </tr>  <tr>  <td>类型：<b>${vo.noticeType}</b></td>  </tr>  <tr>  <td>创建时间：<b>${vo.createDate}</b></td>  </tr>  </table>  <br>  <button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">返回</button>  </form>  </div>  </body>  </html> |

列表与查询页面（notice\_list.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>公告管理</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">公告管理</a>  <br>  <br>  </div>  <br>  <div class="index-content-operation">  <button class="btn btn-default btn-sm" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="window.location.href='notice\_add.jsp'">添加</button>  <div class="index-content-operation-search"><input id="search\_keyword" placeholder="标题" type="text" name="search\_keyword"/><input type="hidden" id="searchColumn" name="searchColumn" value="notice\_name"/><button class="btn btn-primary btn-sm" onclick="searchList()">搜索</button></div>  </div>  <br>  <table class="table table-striped table-hover table-bordered">  <thead>  <tr class="index-content-table-th">  <th>标题</th>  <th>类型</th>  <th>创建时间</th>  <th>操作</th>  </tr>  </thead>  <tbody>  <c:forEach items="${list}" var="vo">  <tr class="index-content-table-td">  <td>${vo.noticeName}</td>  <td>${vo.noticeType}</td>  <td>${vo.createDate}</td>  <td>  <button class="btn btn-success btn-sm" style="padding: 0px 1px;" onclick="window.location.href='NoticeServlet?action=get&id=${vo.id}'">详情</button>&nbsp;  <button class="btn btn-info btn-sm" style="padding: 0px 1px;"  <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if>  onclick="window.location.href='NoticeServlet?action=editPre&id=${vo.id}'">编辑</button>&nbsp;  <button class="btn btn-danger btn-sm" style="padding: 0px 1px;" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="if(window.confirm('将要删除：${vo.noticeName}？'))window.location.href='NoticeServlet?action=delete&id=${vo.id}'">删除</button>  </td>  </tr>  </c:forEach>  </tbody>  </table>  <div style="float: right;padding-right: 10px;color: #515151;"><jsp:include page="split.jsp"/></div>  </div>  </body>  <script>  function searchList() {  window.location.href = "NoticeServlet?action=list&searchColumn="+document.getElementById("searchColumn").value+"&keyword=" + document.getElementById("search\_keyword").value;  }  </script>  </html> |

### 8.3.3 宿管员模块

数据层接口（AdministratorDAO.java）

|  |
| --- |
| package com.demo.dao;  import com.demo.vo.Administrator;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Administrator模块的DAO层（数据层）接口，提供增删改查等数据库操作的方法抽象  \*/  public interface AdministratorDAO {  /\*\*  \* 增加宿管员表记录  \*  \* @param vo  \* @return  \*/  void add(Administrator vo);  /\*\*  \* 根据主键id，删除对应的宿管员表记录  \*  \* @param id  \* @return  \*/  boolean delete(long id);  /\*\*  \* 更新宿管员表记录  \*  \* @param vo  \* @return  \*/  void update(Administrator vo);  /\*\*  \* 根据主键id获取宿管员表记录的详情  \*  \* @param id  \* @return  \*/  Administrator get(Serializable id);  /\*\*  \* 根据条件查询宿管员的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

数据层实现类（AdministratorDAOImpl.java）

|  |
| --- |
| package com.demo.dao.impl;  import com.demo.util.Util;  import com.demo.dao.AdministratorDAO;  import com.demo.vo.Administrator;  import java.io.Serializable;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* Administrator模块的DAO层（数据层）的具体实现类，对AdministratorDAO接口中定义的增删改查等抽象方法作出具体的功能实现  \*/  public class AdministratorDAOImpl implements AdministratorDAO {  //@Override  public void add(Administrator vo) {  String sql = "insert into `t\_administrator` (`administrator\_name`,`administrator\_sex`,`administrator\_phone`,`administrator\_building`,`administrator\_text`) values(?,?,?,?,?)";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getAdministratorName());  ps.setString(2, vo.getAdministratorSex());  ps.setString(3, vo.getAdministratorPhone());  ps.setString(4, vo.getAdministratorBuilding());  ps.setString(5, vo.getAdministratorText());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public void update(Administrator vo) {  String sql = "update `t\_administrator` set `administrator\_name` = ? ,`administrator\_sex` = ? ,`administrator\_phone` = ? ,`administrator\_building` = ? ,`administrator\_text` = ? where `id` = ?";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getAdministratorName());  ps.setString(2, vo.getAdministratorSex());  ps.setString(3, vo.getAdministratorPhone());  ps.setString(4, vo.getAdministratorBuilding());  ps.setString(5, vo.getAdministratorText());  ps.setLong(6, vo.getId());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public boolean delete(long id) {  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "delete from `t\_administrator` where id = " + id;  s.execute(sql);  s.close();  c.close();  return true;  } catch (Exception e) {  e.printStackTrace();  return false;  }  }  //@Override  public Administrator get(Serializable id) {  Administrator vo = null;  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "select \* from `t\_administrator` where id = " + id;  ResultSet rs = s.executeQuery(sql);  if (rs.next()) {  vo = new Administrator();  vo.setId(rs.getLong("id"));  vo.setAdministratorName(rs.getString("administrator\_name"));  vo.setAdministratorSex(rs.getString("administrator\_sex"));  vo.setAdministratorPhone(rs.getString("administrator\_phone"));  vo.setAdministratorBuilding(rs.getString("administrator\_building"));  vo.setAdministratorText(rs.getString("administrator\_text"));  }  c.close();  } catch (Exception e) {  e.printStackTrace();  }  return vo;  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  List<Administrator> list = new ArrayList();  int totalCount = 0;  String condition = "";  String sqlList;  if (params.get("searchColumn") != null && !"".equals(params.get("searchColumn"))) {  condition += " and `" + params.get("searchColumn") + "` like '%" + params.get("keyword") + "%'";  }  try {  Connection c = Util.getConnection();  PreparedStatement ps;  ResultSet rs;  String limit = (params.get("startIndex") != null && params.get("pageSize") != null) ? " limit " + params.get("startIndex") + "," + params.get("pageSize") : "";  sqlList = "select \* from `t\_administrator` where 1=1 " + condition + " order by id asc " + limit + ";";  ps = c.prepareStatement(sqlList);  rs = ps.executeQuery();  while (rs.next()) {  Administrator vo = new Administrator();  vo.setId(rs.getLong("id"));  vo.setAdministratorName(rs.getString("administrator\_name"));  vo.setAdministratorSex(rs.getString("administrator\_sex"));  vo.setAdministratorPhone(rs.getString("administrator\_phone"));  vo.setAdministratorBuilding(rs.getString("administrator\_building"));  vo.setAdministratorText(rs.getString("administrator\_text"));  list.add(vo);  }  String sqlCount = "select count(\*) from `t\_administrator` where 1=1 " + condition;  ps = c.prepareStatement(sqlCount);  rs = ps.executeQuery();  if (rs.next()) {  totalCount = rs.getInt(1);  }  rs.close();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  Map<String, Object> result = new HashMap();  result.put("list", list);  result.put("totalCount", totalCount);  return result;  }  } |

业务层接口（AdministratorService.java）

|  |
| --- |
| package com.demo.service;  import com.demo.vo.Administrator;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Administrator模块的Service层（业务层）接口，提供业务方法的抽象  \*/  public interface AdministratorService {  /\*\*  \* 增加宿管员  \*  \* @param vo  \* @return  \*/  void add(Administrator vo);  /\*\*  \* 删除宿管员  \*  \* @param id  \* @return  \*/  void delete(long id);  /\*\*  \* 修改宿管员  \*  \* @param vo  \* @return  \*/  void update(Administrator vo);  /\*\*  \* 根据主键Id查询宿管员详情  \*  \* @param id  \* @return  \*/  Administrator get(Serializable id);  /\*\*  \* 根据条件查询宿管员的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

业务层实现类（AdministratorServiceImpl.java）

|  |
| --- |
| package com.demo.service.impl;  import com.demo.dao.AdministratorDAO;  import com.demo.dao.impl.AdministratorDAOImpl;  import com.demo.service.AdministratorService;  import com.demo.vo.Administrator;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Administrator模块的Service层（业务层）的具体实现类，对AdministratorService接口中定义的抽象方法作出具体的功能实现  \*/  public class AdministratorServiceImpl implements AdministratorService {  //@Override  public void add(Administrator vo) {  AdministratorDAO administratorDAO = new AdministratorDAOImpl();  administratorDAO.add(vo);  }  //@Override  public void delete(long id) {  AdministratorDAO administratorDAO = new AdministratorDAOImpl();  administratorDAO.delete(id);  }  //@Override  public void update(Administrator vo) {  AdministratorDAO administratorDAO = new AdministratorDAOImpl();  administratorDAO.update(vo);  }  //@Override  public Administrator get(Serializable id) {  AdministratorDAO administratorDAO = new AdministratorDAOImpl();  return administratorDAO.get(id);  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  AdministratorDAO administratorDAO = new AdministratorDAOImpl();  return administratorDAO.list(params);  }  } |

控制层（AdministratorServlet.java）

|  |
| --- |
| package com.demo.servlet;  import com.demo.util.Util;  import com.demo.service.AdministratorService;  import com.demo.service.impl.AdministratorServiceImpl;  import com.demo.vo.Administrator;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  import java.io.IOException;  import java.io.Serializable;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* Administrator模块的Servlet控制层，负责接收页面传过来的请求参数，根据action参数的值来确定页面要执行的具体操作<br>  \* 而后再调用AdministratorService业务层的方法来处理具体的业务，最后将处理完成的结果返回或跳转至相应页面  \*/  //@WebServlet("/AdministratorServlet")  public class AdministratorServlet extends HttpServlet {  /\*\*  \* 处理Post请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //过滤编码  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  String action = Util.decode(request, "action");  if ("add".equals(action)) {//增加  Administrator vo = new Administrator();  //取出页面传进来的各个数据，并设置到Administrator对象的属性里  String administratorName = Util.decode(request, "administratorName");  if (administratorName != null && !administratorName.equals("")) {  vo.setAdministratorName(administratorName);  }  String administratorSex = Util.decode(request, "administratorSex");  if (administratorSex != null && !administratorSex.equals("")) {  vo.setAdministratorSex(administratorSex);  }  String administratorPhone = Util.decode(request, "administratorPhone");  if (administratorPhone != null && !administratorPhone.equals("")) {  vo.setAdministratorPhone(administratorPhone);  }  String administratorBuilding = Util.decode(request, "administratorBuilding");  if (administratorBuilding != null && !administratorBuilding.equals("")) {  vo.setAdministratorBuilding(administratorBuilding);  }  String administratorText = Util.decode(request, "administratorText");  if (administratorText != null && !administratorText.equals("")) {  vo.setAdministratorText(administratorText);  }  AdministratorService administratorService = new AdministratorServiceImpl();  //调用Service层增加方法（add），增加记录  administratorService.add(vo);  this.redirectList(request, response);  } else if ("delete".equals(action)) {//删除  //取出表要删除的宿管员记录的主键  long id = Long.parseLong(Util.decode(request, "id"));  AdministratorService administratorService = new AdministratorServiceImpl();  //调用Service层删除方法（delete），将对应的记录删除  administratorService.delete(id);  this.redirectList(request, response);  } else if ("edit".equals(action)) {//修改  //取出页面传进来的各个数据，并设置到Administrator对象的属性里  Administrator vo = new Administrator();  String id = Util.decode(request, "id");  if (id != null && !id.equals("")) {  vo.setId(Long.valueOf(id));  }  String administratorName = Util.decode(request, "administratorName");  if (administratorName != null && !administratorName.equals("")) {  vo.setAdministratorName(administratorName);  }  String administratorSex = Util.decode(request, "administratorSex");  if (administratorSex != null && !administratorSex.equals("")) {  vo.setAdministratorSex(administratorSex);  }  String administratorPhone = Util.decode(request, "administratorPhone");  if (administratorPhone != null && !administratorPhone.equals("")) {  vo.setAdministratorPhone(administratorPhone);  }  String administratorBuilding = Util.decode(request, "administratorBuilding");  if (administratorBuilding != null && !administratorBuilding.equals("")) {  vo.setAdministratorBuilding(administratorBuilding);  }  String administratorText = Util.decode(request, "administratorText");  if (administratorText != null && !administratorText.equals("")) {  vo.setAdministratorText(administratorText);  }  AdministratorService administratorService = new AdministratorServiceImpl();  //调用Service层更新方法（update），更新记录  administratorService.update(vo);  this.redirectList(request, response);  } else if ("get".equalsIgnoreCase(action) || "editPre".equalsIgnoreCase(action)) {//根据主键ID，查询详情信息并跳转到详情页面或编辑页面  Serializable id = Util.decode(request, "id");//取出页面传入的主键，用于查询详情  AdministratorService administratorService = new AdministratorServiceImpl();  Administrator vo = administratorService.get(id);  request.getSession().setAttribute("vo", vo);  String to = "get".equalsIgnoreCase(action) ? "info" : "edit";//判断是去详情显示页面还是编辑页面  response.sendRedirect("administrator\_" + to + ".jsp");  } else {//默认去列表页面  this.redirectList(request, response);  }  }  /\*\*  \* 处理Get请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doPost(request, response);//Get请求和Post请求的处理是一样的，所以把request、response转交给Post方法就好  }  /\*\*  \* 根据参数，查询出条例条件的记录集合，最后将数据返回给调用处或者将数据集合设置到session域里，再跳转到对应的列表页面  \*  \* @param request  \* @param response  \*/  private void redirectList(HttpServletRequest request, HttpServletResponse response) throws IOException {  //查询列和关键字  String searchColumn = Util.decode(request, "searchColumn");  String keyword = Util.decode(request, "keyword");  Map<String, Object> params = new HashMap();//用来保存控制层传进来的参数(查询条件)  params.put("searchColumn", searchColumn);//要查询的列  params.put("keyword", keyword);//查询的关键字  AdministratorService administratorService = new AdministratorServiceImpl();  Map<String, Object> map = administratorService.list(params);  request.getSession().setAttribute("list", map.get("list"));  Integer totalRecord = (Integer) map.get("totalCount");//根据查询条件取出对应的总记录数，用于分页  String pageNum = Util.decode(request, "pageNum");//封装分页参数  com.demo.util.PageBean<Object> pb = new com.demo.util.PageBean(Integer.valueOf(pageNum != null ? pageNum : "1"), totalRecord);  params.put("startIndex", pb.getStartIndex());  params.put("pageSize", pb.getPageSize());  List list = (List) administratorService.list(params).get("list");//根据分页参数startIndex、pageSize查询出来的最终结果list  pb.setServlet("AdministratorServlet");  pb.setSearchColumn(searchColumn);  pb.setKeyword(keyword);  pb.setList(list);  request.getSession().setAttribute("pageBean", pb);  request.getSession().setAttribute("list", pb.getList());  response.sendRedirect("administrator\_list.jsp");  }  } |

JavaBean实体模型层定义（Administrator.java）

|  |
| --- |
| package com.demo.vo;  import java.io.Serializable;  /\*\*  \* 宿管员（t\_administrator表对应的Java实体类）  \*/  public class Administrator implements Serializable {  private Long id;//主键  private String administratorName;//姓名  private String administratorSex;//性别:男/女  private String administratorPhone;//电话  private String administratorBuilding;//楼栋  private String administratorText;//备注  public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public String getAdministratorName() {  return administratorName;  }  public void setAdministratorName(String administratorName) {  this.administratorName = administratorName;  }  public String getAdministratorSex() {  return administratorSex;  }  public void setAdministratorSex(String administratorSex) {  this.administratorSex = administratorSex;  }  public String getAdministratorPhone() {  return administratorPhone;  }  public void setAdministratorPhone(String administratorPhone) {  this.administratorPhone = administratorPhone;  }  public String getAdministratorBuilding() {  return administratorBuilding;  }  public void setAdministratorBuilding(String administratorBuilding) {  this.administratorBuilding = administratorBuilding;  }  public String getAdministratorText() {  return administratorText;  }  public void setAdministratorText(String administratorText) {  this.administratorText = administratorText;  }  } |

增加信息页面（administrator\_add.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>添加宿管员</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">添加宿管员</a>  <br>  <br>  </div>  <br>  <form action="AdministratorServlet?action=add" method="post" onsubmit="return check()">  <table class="index-content-table-add">  <tr>  <td width="12%">姓名：</td><td><input class="index-content-table-td-add" type="text" id="administratorName" name="administratorName" value=""/></td>  </tr>  <tr>  <td width="12%">性别：</td>  <td>  <input name="administratorSex" type="radio" value="男" checked="checked"/>&nbsp;&nbsp;&nbsp;男&nbsp;&nbsp;&nbsp;&nbsp;  <input name="administratorSex" type="radio" value="女"/>&nbsp;&nbsp;&nbsp;女&nbsp;&nbsp;&nbsp;&nbsp;  </td>  </tr>  <tr>  <td width="12%">电话：</td><td><input class="index-content-table-td-add" type="text" id="administratorPhone" name="administratorPhone" value=""/></td>  </tr>  <tr>  <td width="12%">楼栋：</td><td><input class="index-content-table-td-add" type="text" id="administratorBuilding" name="administratorBuilding" value=""/></td>  </tr>  <tr>  <td width="12%">备注：</td><td><textarea id="administratorText" name="administratorText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......"></textarea></td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("administratorName").value.trim().length == 0) {  alert("姓名不能为空!");  return false;  }  if (document.getElementById("administratorPhone").value.trim().length == 0) {  alert("电话不能为空!");  return false;  }  if (document.getElementById("administratorBuilding").value.trim().length == 0) {  alert("楼栋不能为空!");  return false;  }  return true;  }  </script>  </html> |

更新信息页面（administrator\_edit.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>修改宿管员</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">修改宿管员</a>  <br>  <br>  </div>  <br>  <form action="AdministratorServlet?action=edit" method="post" onsubmit="return check()">  <input type="hidden" id="id" name="id" value="${vo.id}"/>    <table class="index-content-table-add">  <tr>  <td width="12%">姓名：</td><td><input class="index-content-table-td-add" type="text" id="administratorName" name="administratorName" value="${vo.administratorName}"/></td>  </tr>  <tr>  <td width="12%">性别：</td>  <td>  <input name="administratorSex" type="radio" value="男" ${vo.administratorSex=='男'?'checked':''}/>&nbsp;&nbsp;&nbsp;男&nbsp;&nbsp;&nbsp;&nbsp;  <input name="administratorSex" type="radio" value="女" ${vo.administratorSex=='女'?'checked':''}/>&nbsp;&nbsp;&nbsp;女&nbsp;&nbsp;&nbsp;&nbsp;  </td>  </tr>  <tr>  <td width="12%">电话：</td><td><input class="index-content-table-td-add" type="text" id="administratorPhone" name="administratorPhone" value="${vo.administratorPhone}"/></td>  </tr>  <tr>  <td width="12%">楼栋：</td><td><input class="index-content-table-td-add" type="text" id="administratorBuilding" name="administratorBuilding" value="${vo.administratorBuilding}"/></td>  </tr>  <tr>  <td width="12%">备注：</td><td><textarea id="administratorText" name="administratorText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......">${vo.administratorText}</textarea></td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("administratorName").value.trim().length == 0) {  alert("姓名不能为空!");  return false;  }  if (document.getElementById("administratorPhone").value.trim().length == 0) {  alert("电话不能为空!");  return false;  }  if (document.getElementById("administratorBuilding").value.trim().length == 0) {  alert("楼栋不能为空!");  return false;  }  return true;  }  </script>  </html> |

查看详情页面（administrator\_info.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>宿管员详情</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">宿管员详情</a>  <br>  <br>  </div>  <br>  <form>  <table class="index-content-table-add" style="font-size: 18px;">  <tr>  <td>姓名：<b>${vo.administratorName}</b></td>  </tr>  <tr>  <td>性别：  <b>${vo.administratorSex}</b>  </td>  </tr>  <tr>  <td>电话：<b>${vo.administratorPhone}</b></td>  </tr>  <tr>  <td>楼栋：<b>${vo.administratorBuilding}</b></td>  </tr>  <tr>  <td>备注：<b>${vo.administratorText}</b></td>  </tr>  </table>  <br>  <button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">返回</button>  </form>  </div>  </body>  </html> |

列表与查询页面（administrator\_list.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>宿管员管理</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">宿管员管理</a>  <br>  <br>  </div>  <br>  <div class="index-content-operation">  <button class="btn btn-default btn-sm" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="window.location.href='administrator\_add.jsp'">添加</button>  <div class="index-content-operation-search"><input id="search\_keyword" placeholder="姓名" type="text" name="search\_keyword"/><input type="hidden" id="searchColumn" name="searchColumn" value="administrator\_name"/><button class="btn btn-primary btn-sm" onclick="searchList()">搜索</button></div>  </div>  <br>  <table class="table table-striped table-hover table-bordered">  <thead>  <tr class="index-content-table-th">  <th>姓名</th>  <th>性别</th>  <th>电话</th>  <th>楼栋</th>  <th>操作</th>  </tr>  </thead>  <tbody>  <c:forEach items="${list}" var="vo">  <tr class="index-content-table-td">  <td>${vo.administratorName}</td>  <td>${vo.administratorSex}</td>  <td>${vo.administratorPhone}</td>  <td>${vo.administratorBuilding}</td>  <td>  <button class="btn btn-success btn-sm" style="padding: 0px 1px;" onclick="window.location.href='AdministratorServlet?action=get&id=${vo.id}'">详情</button>&nbsp;  <button class="btn btn-info btn-sm" style="padding: 0px 1px;"  <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if>  onclick="window.location.href='AdministratorServlet?action=editPre&id=${vo.id}'">编辑</button>&nbsp;  <button class="btn btn-danger btn-sm" style="padding: 0px 1px;" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="if(window.confirm('将要删除：${vo.administratorName}？'))window.location.href='AdministratorServlet?action=delete&id=${vo.id}'">删除</button>  </td>  </tr>  </c:forEach>  </tbody>  </table>  <div style="float: right;padding-right: 10px;color: #515151;"><jsp:include page="split.jsp"/></div>  </div>  </body>  <script>  function searchList() {  window.location.href = "AdministratorServlet?action=list&searchColumn="+document.getElementById("searchColumn").value+"&keyword=" + document.getElementById("search\_keyword").value;  }  </script>  </html> |

### 8.3.4 宿舍模块

数据层接口（DormitoryDAO.java）

|  |
| --- |
| package com.demo.dao;  import com.demo.vo.Dormitory;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Dormitory模块的DAO层（数据层）接口，提供增删改查等数据库操作的方法抽象  \*/  public interface DormitoryDAO {  /\*\*  \* 增加宿舍表记录  \*  \* @param vo  \* @return  \*/  void add(Dormitory vo);  /\*\*  \* 根据主键id，删除对应的宿舍表记录  \*  \* @param id  \* @return  \*/  boolean delete(long id);  /\*\*  \* 更新宿舍表记录  \*  \* @param vo  \* @return  \*/  void update(Dormitory vo);  /\*\*  \* 根据主键id获取宿舍表记录的详情  \*  \* @param id  \* @return  \*/  Dormitory get(Serializable id);  /\*\*  \* 根据条件查询宿舍的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

数据层实现类（DormitoryDAOImpl.java）

|  |
| --- |
| package com.demo.dao.impl;  import com.demo.util.Util;  import com.demo.dao.DormitoryDAO;  import com.demo.vo.Dormitory;  import java.io.Serializable;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* Dormitory模块的DAO层（数据层）的具体实现类，对DormitoryDAO接口中定义的增删改查等抽象方法作出具体的功能实现  \*/  public class DormitoryDAOImpl implements DormitoryDAO {  //@Override  public void add(Dormitory vo) {  String sql = "insert into `t\_dormitory` (`dormitory\_name`,`dormitory\_building`,`dormitory\_bedcount`,`dormitory\_administrator`,`dormitory\_text`) values(?,?,?,?,?)";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getDormitoryName());  ps.setString(2, vo.getDormitoryBuilding());  ps.setString(3, vo.getDormitoryBedcount());  ps.setString(4, vo.getDormitoryAdministrator());  ps.setString(5, vo.getDormitoryText());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public void update(Dormitory vo) {  String sql = "update `t\_dormitory` set `dormitory\_name` = ? ,`dormitory\_building` = ? ,`dormitory\_bedcount` = ? ,`dormitory\_administrator` = ? ,`dormitory\_text` = ? where `id` = ?";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getDormitoryName());  ps.setString(2, vo.getDormitoryBuilding());  ps.setString(3, vo.getDormitoryBedcount());  ps.setString(4, vo.getDormitoryAdministrator());  ps.setString(5, vo.getDormitoryText());  ps.setLong(6, vo.getId());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public boolean delete(long id) {  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "delete from `t\_dormitory` where id = " + id;  s.execute(sql);  s.close();  c.close();  return true;  } catch (Exception e) {  e.printStackTrace();  return false;  }  }  //@Override  public Dormitory get(Serializable id) {  Dormitory vo = null;  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "select \* from `t\_dormitory` where id = " + id;  ResultSet rs = s.executeQuery(sql);  if (rs.next()) {  vo = new Dormitory();  vo.setId(rs.getLong("id"));  vo.setDormitoryName(rs.getString("dormitory\_name"));  vo.setDormitoryBuilding(rs.getString("dormitory\_building"));  vo.setDormitoryBedcount(rs.getString("dormitory\_bedcount"));  vo.setDormitoryAdministrator(rs.getString("dormitory\_administrator"));  vo.setDormitoryText(rs.getString("dormitory\_text"));  }  c.close();  } catch (Exception e) {  e.printStackTrace();  }  return vo;  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  List<Dormitory> list = new ArrayList();  int totalCount = 0;  String condition = "";  String sqlList;  if (params.get("searchColumn") != null && !"".equals(params.get("searchColumn"))) {  condition += " and `" + params.get("searchColumn") + "` like '%" + params.get("keyword") + "%'";  }  try {  Connection c = Util.getConnection();  PreparedStatement ps;  ResultSet rs;  String limit = (params.get("startIndex") != null && params.get("pageSize") != null) ? " limit " + params.get("startIndex") + "," + params.get("pageSize") : "";  sqlList = "select \* from `t\_dormitory` where 1=1 " + condition + " order by id asc " + limit + ";";  ps = c.prepareStatement(sqlList);  rs = ps.executeQuery();  while (rs.next()) {  Dormitory vo = new Dormitory();  vo.setId(rs.getLong("id"));  vo.setDormitoryName(rs.getString("dormitory\_name"));  vo.setDormitoryBuilding(rs.getString("dormitory\_building"));  vo.setDormitoryBedcount(rs.getString("dormitory\_bedcount"));  vo.setDormitoryAdministrator(rs.getString("dormitory\_administrator"));  vo.setDormitoryText(rs.getString("dormitory\_text"));  list.add(vo);  }  String sqlCount = "select count(\*) from `t\_dormitory` where 1=1 " + condition;  ps = c.prepareStatement(sqlCount);  rs = ps.executeQuery();  if (rs.next()) {  totalCount = rs.getInt(1);  }  rs.close();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  Map<String, Object> result = new HashMap();  result.put("list", list);  result.put("totalCount", totalCount);  return result;  }  } |

业务层接口（DormitoryService.java）

|  |
| --- |
| package com.demo.service;  import com.demo.vo.Dormitory;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Dormitory模块的Service层（业务层）接口，提供业务方法的抽象  \*/  public interface DormitoryService {  /\*\*  \* 增加宿舍  \*  \* @param vo  \* @return  \*/  void add(Dormitory vo);  /\*\*  \* 删除宿舍  \*  \* @param id  \* @return  \*/  void delete(long id);  /\*\*  \* 修改宿舍  \*  \* @param vo  \* @return  \*/  void update(Dormitory vo);  /\*\*  \* 根据主键Id查询宿舍详情  \*  \* @param id  \* @return  \*/  Dormitory get(Serializable id);  /\*\*  \* 根据条件查询宿舍的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

业务层实现类（DormitoryServiceImpl.java）

|  |
| --- |
| package com.demo.service.impl;  import com.demo.dao.DormitoryDAO;  import com.demo.dao.impl.DormitoryDAOImpl;  import com.demo.service.DormitoryService;  import com.demo.vo.Dormitory;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Dormitory模块的Service层（业务层）的具体实现类，对DormitoryService接口中定义的抽象方法作出具体的功能实现  \*/  public class DormitoryServiceImpl implements DormitoryService {  //@Override  public void add(Dormitory vo) {  DormitoryDAO dormitoryDAO = new DormitoryDAOImpl();  dormitoryDAO.add(vo);  }  //@Override  public void delete(long id) {  DormitoryDAO dormitoryDAO = new DormitoryDAOImpl();  dormitoryDAO.delete(id);  }  //@Override  public void update(Dormitory vo) {  DormitoryDAO dormitoryDAO = new DormitoryDAOImpl();  dormitoryDAO.update(vo);  }  //@Override  public Dormitory get(Serializable id) {  DormitoryDAO dormitoryDAO = new DormitoryDAOImpl();  return dormitoryDAO.get(id);  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  DormitoryDAO dormitoryDAO = new DormitoryDAOImpl();  return dormitoryDAO.list(params);  }  } |

控制层（DormitoryServlet.java）

|  |
| --- |
| package com.demo.servlet;  import com.demo.util.Util;  import com.demo.service.DormitoryService;  import com.demo.service.impl.DormitoryServiceImpl;  import com.demo.vo.Dormitory;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  import java.io.IOException;  import java.io.Serializable;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* Dormitory模块的Servlet控制层，负责接收页面传过来的请求参数，根据action参数的值来确定页面要执行的具体操作<br>  \* 而后再调用DormitoryService业务层的方法来处理具体的业务，最后将处理完成的结果返回或跳转至相应页面  \*/  //@WebServlet("/DormitoryServlet")  public class DormitoryServlet extends HttpServlet {  /\*\*  \* 处理Post请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //过滤编码  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  String action = Util.decode(request, "action");  if ("add".equals(action)) {//增加  Dormitory vo = new Dormitory();  //取出页面传进来的各个数据，并设置到Dormitory对象的属性里  String dormitoryName = Util.decode(request, "dormitoryName");  if (dormitoryName != null && !dormitoryName.equals("")) {  vo.setDormitoryName(dormitoryName);  }  String dormitoryBuilding = Util.decode(request, "dormitoryBuilding");  if (dormitoryBuilding != null && !dormitoryBuilding.equals("")) {  vo.setDormitoryBuilding(dormitoryBuilding);  }  String dormitoryBedcount = Util.decode(request, "dormitoryBedcount");  if (dormitoryBedcount != null && !dormitoryBedcount.equals("")) {  vo.setDormitoryBedcount(dormitoryBedcount);  }  String dormitoryAdministrator = Util.decode(request, "dormitoryAdministrator");  if (dormitoryAdministrator != null && !dormitoryAdministrator.equals("")) {  vo.setDormitoryAdministrator(dormitoryAdministrator);  }  String dormitoryText = Util.decode(request, "dormitoryText");  if (dormitoryText != null && !dormitoryText.equals("")) {  vo.setDormitoryText(dormitoryText);  }  DormitoryService dormitoryService = new DormitoryServiceImpl();  //调用Service层增加方法（add），增加记录  dormitoryService.add(vo);  this.redirectList(request, response);  } else if ("delete".equals(action)) {//删除  //取出表要删除的宿舍记录的主键  long id = Long.parseLong(Util.decode(request, "id"));  DormitoryService dormitoryService = new DormitoryServiceImpl();  //调用Service层删除方法（delete），将对应的记录删除  dormitoryService.delete(id);  this.redirectList(request, response);  } else if ("edit".equals(action)) {//修改  //取出页面传进来的各个数据，并设置到Dormitory对象的属性里  Dormitory vo = new Dormitory();  String id = Util.decode(request, "id");  if (id != null && !id.equals("")) {  vo.setId(Long.valueOf(id));  }  String dormitoryName = Util.decode(request, "dormitoryName");  if (dormitoryName != null && !dormitoryName.equals("")) {  vo.setDormitoryName(dormitoryName);  }  String dormitoryBuilding = Util.decode(request, "dormitoryBuilding");  if (dormitoryBuilding != null && !dormitoryBuilding.equals("")) {  vo.setDormitoryBuilding(dormitoryBuilding);  }  String dormitoryBedcount = Util.decode(request, "dormitoryBedcount");  if (dormitoryBedcount != null && !dormitoryBedcount.equals("")) {  vo.setDormitoryBedcount(dormitoryBedcount);  }  String dormitoryAdministrator = Util.decode(request, "dormitoryAdministrator");  if (dormitoryAdministrator != null && !dormitoryAdministrator.equals("")) {  vo.setDormitoryAdministrator(dormitoryAdministrator);  }  String dormitoryText = Util.decode(request, "dormitoryText");  if (dormitoryText != null && !dormitoryText.equals("")) {  vo.setDormitoryText(dormitoryText);  }  DormitoryService dormitoryService = new DormitoryServiceImpl();  //调用Service层更新方法（update），更新记录  dormitoryService.update(vo);  this.redirectList(request, response);  } else if ("get".equalsIgnoreCase(action) || "editPre".equalsIgnoreCase(action)) {//根据主键ID，查询详情信息并跳转到详情页面或编辑页面  Serializable id = Util.decode(request, "id");//取出页面传入的主键，用于查询详情  DormitoryService dormitoryService = new DormitoryServiceImpl();  Dormitory vo = dormitoryService.get(id);  request.getSession().setAttribute("vo", vo);  String to = "get".equalsIgnoreCase(action) ? "info" : "edit";//判断是去详情显示页面还是编辑页面  response.sendRedirect("dormitory\_" + to + ".jsp");  } else {//默认去列表页面  this.redirectList(request, response);  }  }  /\*\*  \* 处理Get请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doPost(request, response);//Get请求和Post请求的处理是一样的，所以把request、response转交给Post方法就好  }  /\*\*  \* 根据参数，查询出条例条件的记录集合，最后将数据返回给调用处或者将数据集合设置到session域里，再跳转到对应的列表页面  \*  \* @param request  \* @param response  \*/  private void redirectList(HttpServletRequest request, HttpServletResponse response) throws IOException {  //查询列和关键字  String searchColumn = Util.decode(request, "searchColumn");  String keyword = Util.decode(request, "keyword");  Map<String, Object> params = new HashMap();//用来保存控制层传进来的参数(查询条件)  params.put("searchColumn", searchColumn);//要查询的列  params.put("keyword", keyword);//查询的关键字  DormitoryService dormitoryService = new DormitoryServiceImpl();  Map<String, Object> map = dormitoryService.list(params);  request.getSession().setAttribute("list", map.get("list"));  Integer totalRecord = (Integer) map.get("totalCount");//根据查询条件取出对应的总记录数，用于分页  String pageNum = Util.decode(request, "pageNum");//封装分页参数  com.demo.util.PageBean<Object> pb = new com.demo.util.PageBean(Integer.valueOf(pageNum != null ? pageNum : "1"), totalRecord);  params.put("startIndex", pb.getStartIndex());  params.put("pageSize", pb.getPageSize());  List list = (List) dormitoryService.list(params).get("list");//根据分页参数startIndex、pageSize查询出来的最终结果list  pb.setServlet("DormitoryServlet");  pb.setSearchColumn(searchColumn);  pb.setKeyword(keyword);  pb.setList(list);  request.getSession().setAttribute("pageBean", pb);  request.getSession().setAttribute("list", pb.getList());  response.sendRedirect("dormitory\_list.jsp");  }  } |

JavaBean实体模型层定义（Dormitory.java）

|  |
| --- |
| package com.demo.vo;  import java.io.Serializable;  /\*\*  \* 宿舍（t\_dormitory表对应的Java实体类）  \*/  public class Dormitory implements Serializable {  private Long id;//主键  private String dormitoryName;//宿舍  private String dormitoryBuilding;//楼栋  private String dormitoryBedcount;//床位数  private String dormitoryAdministrator;//宿管员  private String dormitoryText;//备注  public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public String getDormitoryName() {  return dormitoryName;  }  public void setDormitoryName(String dormitoryName) {  this.dormitoryName = dormitoryName;  }  public String getDormitoryBuilding() {  return dormitoryBuilding;  }  public void setDormitoryBuilding(String dormitoryBuilding) {  this.dormitoryBuilding = dormitoryBuilding;  }  public String getDormitoryBedcount() {  return dormitoryBedcount;  }  public void setDormitoryBedcount(String dormitoryBedcount) {  this.dormitoryBedcount = dormitoryBedcount;  }  public String getDormitoryAdministrator() {  return dormitoryAdministrator;  }  public void setDormitoryAdministrator(String dormitoryAdministrator) {  this.dormitoryAdministrator = dormitoryAdministrator;  }  public String getDormitoryText() {  return dormitoryText;  }  public void setDormitoryText(String dormitoryText) {  this.dormitoryText = dormitoryText;  }  } |

增加信息页面（dormitory\_add.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>添加宿舍</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">添加宿舍</a>  <br>  <br>  </div>  <br>  <form action="DormitoryServlet?action=add" method="post" onsubmit="return check()">  <table class="index-content-table-add">  <tr>  <td width="12%">宿舍：</td><td><input class="index-content-table-td-add" type="text" id="dormitoryName" name="dormitoryName" value=""/></td>  </tr>  <tr>  <td width="12%">楼栋：</td><td><input class="index-content-table-td-add" type="text" id="dormitoryBuilding" name="dormitoryBuilding" value=""/></td>  </tr>  <tr>  <td width="12%">床位数：</td><td><input class="index-content-table-td-add" type="text" id="dormitoryBedcount" name="dormitoryBedcount" value=""/></td>  </tr>  <tr>  <td width="12%">宿管员：</td><td><input class="index-content-table-td-add" type="text" id="dormitoryAdministrator" name="dormitoryAdministrator" value=""/></td>  </tr>  <tr>  <td width="12%">备注：</td><td><textarea id="dormitoryText" name="dormitoryText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......"></textarea></td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("dormitoryName").value.trim().length == 0) {  alert("宿舍不能为空!");  return false;  }  if (document.getElementById("dormitoryBuilding").value.trim().length == 0) {  alert("楼栋不能为空!");  return false;  }  if (document.getElementById("dormitoryBedcount").value.trim().length == 0) {  alert("床位数不能为空!");  return false;  }  if (document.getElementById("dormitoryAdministrator").value.trim().length == 0) {  alert("宿管员不能为空!");  return false;  }  return true;  }  </script>  </html> |

更新信息页面（dormitory\_edit.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>修改宿舍</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">修改宿舍</a>  <br>  <br>  </div>  <br>  <form action="DormitoryServlet?action=edit" method="post" onsubmit="return check()">  <input type="hidden" id="id" name="id" value="${vo.id}"/>    <table class="index-content-table-add">  <tr>  <td width="12%">宿舍：</td><td><input class="index-content-table-td-add" type="text" id="dormitoryName" name="dormitoryName" value="${vo.dormitoryName}"/></td>  </tr>  <tr>  <td width="12%">楼栋：</td><td><input class="index-content-table-td-add" type="text" id="dormitoryBuilding" name="dormitoryBuilding" value="${vo.dormitoryBuilding}"/></td>  </tr>  <tr>  <td width="12%">床位数：</td><td><input class="index-content-table-td-add" type="text" id="dormitoryBedcount" name="dormitoryBedcount" value="${vo.dormitoryBedcount}"/></td>  </tr>  <tr>  <td width="12%">宿管员：</td><td><input class="index-content-table-td-add" type="text" id="dormitoryAdministrator" name="dormitoryAdministrator" value="${vo.dormitoryAdministrator}"/></td>  </tr>  <tr>  <td width="12%">备注：</td><td><textarea id="dormitoryText" name="dormitoryText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......">${vo.dormitoryText}</textarea></td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("dormitoryName").value.trim().length == 0) {  alert("宿舍不能为空!");  return false;  }  if (document.getElementById("dormitoryBuilding").value.trim().length == 0) {  alert("楼栋不能为空!");  return false;  }  if (document.getElementById("dormitoryBedcount").value.trim().length == 0) {  alert("床位数不能为空!");  return false;  }  if (document.getElementById("dormitoryAdministrator").value.trim().length == 0) {  alert("宿管员不能为空!");  return false;  }  return true;  }  </script>  </html> |

查看详情页面（dormitory\_info.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>宿舍详情</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">宿舍详情</a>  <br>  <br>  </div>  <br>  <form>  <table class="index-content-table-add" style="font-size: 18px;">  <tr>  <td>宿舍：<b>${vo.dormitoryName}</b></td>  </tr>  <tr>  <td>楼栋：<b>${vo.dormitoryBuilding}</b></td>  </tr>  <tr>  <td>床位数：<b>${vo.dormitoryBedcount}</b></td>  </tr>  <tr>  <td>宿管员：<b>${vo.dormitoryAdministrator}</b></td>  </tr>  <tr>  <td>备注：<b>${vo.dormitoryText}</b></td>  </tr>  </table>  <br>  <button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">返回</button>  </form>  </div>  </body>  </html> |

列表与查询页面（dormitory\_list.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>宿舍管理</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">宿舍管理</a>  <br>  <br>  </div>  <br>  <div class="index-content-operation">  <button class="btn btn-default btn-sm" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="window.location.href='dormitory\_add.jsp'">添加</button>  <div class="index-content-operation-search"><input id="search\_keyword" placeholder="宿舍" type="text" name="search\_keyword"/><input type="hidden" id="searchColumn" name="searchColumn" value="dormitory\_name"/><button class="btn btn-primary btn-sm" onclick="searchList()">搜索</button></div>  </div>  <br>  <table class="table table-striped table-hover table-bordered">  <thead>  <tr class="index-content-table-th">  <th>宿舍</th>  <th>楼栋</th>  <th>床位数</th>  <th>宿管员</th>  <th>操作</th>  </tr>  </thead>  <tbody>  <c:forEach items="${list}" var="vo">  <tr class="index-content-table-td">  <td>${vo.dormitoryName}</td>  <td>${vo.dormitoryBuilding}</td>  <td>${vo.dormitoryBedcount}</td>  <td>${vo.dormitoryAdministrator}</td>  <td>  <button class="btn btn-success btn-sm" style="padding: 0px 1px;" onclick="window.location.href='DormitoryServlet?action=get&id=${vo.id}'">详情</button>&nbsp;  <button class="btn btn-info btn-sm" style="padding: 0px 1px;"  <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if>  onclick="window.location.href='DormitoryServlet?action=editPre&id=${vo.id}'">编辑</button>&nbsp;  <button class="btn btn-danger btn-sm" style="padding: 0px 1px;" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="if(window.confirm('将要删除：${vo.dormitoryName}？'))window.location.href='DormitoryServlet?action=delete&id=${vo.id}'">删除</button>  </td>  </tr>  </c:forEach>  </tbody>  </table>  <div style="float: right;padding-right: 10px;color: #515151;"><jsp:include page="split.jsp"/></div>  </div>  </body>  <script>  function searchList() {  window.location.href = "DormitoryServlet?action=list&searchColumn="+document.getElementById("searchColumn").value+"&keyword=" + document.getElementById("search\_keyword").value;  }  </script>  </html> |

### 8.3.5 学生模块

数据层接口（StudentDAO.java）

|  |
| --- |
| package com.demo.dao;  import com.demo.vo.Student;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Student模块的DAO层（数据层）接口，提供增删改查等数据库操作的方法抽象  \*/  public interface StudentDAO {  /\*\*  \* 增加学生表记录  \*  \* @param vo  \* @return  \*/  void add(Student vo);  /\*\*  \* 根据主键id，删除对应的学生表记录  \*  \* @param id  \* @return  \*/  boolean delete(long id);  /\*\*  \* 更新学生表记录  \*  \* @param vo  \* @return  \*/  void update(Student vo);  /\*\*  \* 根据主键id获取学生表记录的详情  \*  \* @param id  \* @return  \*/  Student get(Serializable id);  /\*\*  \* 根据条件查询学生的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

数据层实现类（StudentDAOImpl.java）

|  |
| --- |
| package com.demo.dao.impl;  import com.demo.util.Util;  import com.demo.dao.StudentDAO;  import com.demo.vo.Student;  import java.io.Serializable;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* Student模块的DAO层（数据层）的具体实现类，对StudentDAO接口中定义的增删改查等抽象方法作出具体的功能实现  \*/  public class StudentDAOImpl implements StudentDAO {  //@Override  public void add(Student vo) {  String sql = "insert into `t\_student` (`student\_name`,`student\_sex`,`student\_no`,`student\_building`,`student\_dormitory`,`student\_phone`,`student\_text`) values(?,?,?,?,?,?,?)";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getStudentName());  ps.setString(2, vo.getStudentSex());  ps.setString(3, vo.getStudentNo());  ps.setString(4, vo.getStudentBuilding());  ps.setString(5, vo.getStudentDormitory());  ps.setString(6, vo.getStudentPhone());  ps.setString(7, vo.getStudentText());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public void update(Student vo) {  String sql = "update `t\_student` set `student\_name` = ? ,`student\_sex` = ? ,`student\_no` = ? ,`student\_building` = ? ,`student\_dormitory` = ? ,`student\_phone` = ? ,`student\_text` = ? where `id` = ?";  try {  Connection c = Util.getConnection();  PreparedStatement ps = c.prepareStatement(sql);    ps.setString(1, vo.getStudentName());  ps.setString(2, vo.getStudentSex());  ps.setString(3, vo.getStudentNo());  ps.setString(4, vo.getStudentBuilding());  ps.setString(5, vo.getStudentDormitory());  ps.setString(6, vo.getStudentPhone());  ps.setString(7, vo.getStudentText());  ps.setLong(8, vo.getId());  ps.execute();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  }  //@Override  public boolean delete(long id) {  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "delete from `t\_student` where id = " + id;  s.execute(sql);  s.close();  c.close();  return true;  } catch (Exception e) {  e.printStackTrace();  return false;  }  }  //@Override  public Student get(Serializable id) {  Student vo = null;  try {  Connection c = Util.getConnection();  Statement s = c.createStatement();  String sql = "select \* from `t\_student` where id = " + id;  ResultSet rs = s.executeQuery(sql);  if (rs.next()) {  vo = new Student();  vo.setId(rs.getLong("id"));  vo.setStudentName(rs.getString("student\_name"));  vo.setStudentSex(rs.getString("student\_sex"));  vo.setStudentNo(rs.getString("student\_no"));  vo.setStudentBuilding(rs.getString("student\_building"));  vo.setStudentDormitory(rs.getString("student\_dormitory"));  vo.setStudentPhone(rs.getString("student\_phone"));  vo.setStudentText(rs.getString("student\_text"));  }  c.close();  } catch (Exception e) {  e.printStackTrace();  }  return vo;  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  List<Student> list = new ArrayList();  int totalCount = 0;  String condition = "";  String sqlList;  if (params.get("searchColumn") != null && !"".equals(params.get("searchColumn"))) {  condition += " and `" + params.get("searchColumn") + "` like '%" + params.get("keyword") + "%'";  }  try {  Connection c = Util.getConnection();  PreparedStatement ps;  ResultSet rs;  String limit = (params.get("startIndex") != null && params.get("pageSize") != null) ? " limit " + params.get("startIndex") + "," + params.get("pageSize") : "";  sqlList = "select \* from `t\_student` where 1=1 " + condition + " order by id asc " + limit + ";";  ps = c.prepareStatement(sqlList);  rs = ps.executeQuery();  while (rs.next()) {  Student vo = new Student();  vo.setId(rs.getLong("id"));  vo.setStudentName(rs.getString("student\_name"));  vo.setStudentSex(rs.getString("student\_sex"));  vo.setStudentNo(rs.getString("student\_no"));  vo.setStudentBuilding(rs.getString("student\_building"));  vo.setStudentDormitory(rs.getString("student\_dormitory"));  vo.setStudentPhone(rs.getString("student\_phone"));  vo.setStudentText(rs.getString("student\_text"));  list.add(vo);  }  String sqlCount = "select count(\*) from `t\_student` where 1=1 " + condition;  ps = c.prepareStatement(sqlCount);  rs = ps.executeQuery();  if (rs.next()) {  totalCount = rs.getInt(1);  }  rs.close();  ps.close();  c.close();  } catch (Exception e) {  e.printStackTrace();  }  Map<String, Object> result = new HashMap();  result.put("list", list);  result.put("totalCount", totalCount);  return result;  }  } |

业务层接口（StudentService.java）

|  |
| --- |
| package com.demo.service;  import com.demo.vo.Student;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Student模块的Service层（业务层）接口，提供业务方法的抽象  \*/  public interface StudentService {  /\*\*  \* 增加学生  \*  \* @param vo  \* @return  \*/  void add(Student vo);  /\*\*  \* 删除学生  \*  \* @param id  \* @return  \*/  void delete(long id);  /\*\*  \* 修改学生  \*  \* @param vo  \* @return  \*/  void update(Student vo);  /\*\*  \* 根据主键Id查询学生详情  \*  \* @param id  \* @return  \*/  Student get(Serializable id);  /\*\*  \* 根据条件查询学生的列表与数量  \*  \* @param params  \* @return  \*/  Map<String, Object> list(Map<String, Object> params);  } |

业务层实现类（StudentServiceImpl.java）

|  |
| --- |
| package com.demo.service.impl;  import com.demo.dao.StudentDAO;  import com.demo.dao.impl.StudentDAOImpl;  import com.demo.service.StudentService;  import com.demo.vo.Student;  import java.io.Serializable;  import java.util.Map;  /\*\*  \* Student模块的Service层（业务层）的具体实现类，对StudentService接口中定义的抽象方法作出具体的功能实现  \*/  public class StudentServiceImpl implements StudentService {  //@Override  public void add(Student vo) {  StudentDAO studentDAO = new StudentDAOImpl();  studentDAO.add(vo);  }  //@Override  public void delete(long id) {  StudentDAO studentDAO = new StudentDAOImpl();  studentDAO.delete(id);  }  //@Override  public void update(Student vo) {  StudentDAO studentDAO = new StudentDAOImpl();  studentDAO.update(vo);  }  //@Override  public Student get(Serializable id) {  StudentDAO studentDAO = new StudentDAOImpl();  return studentDAO.get(id);  }  //@Override  public Map<String, Object> list(Map<String, Object> params) {  StudentDAO studentDAO = new StudentDAOImpl();  return studentDAO.list(params);  }  } |

控制层（StudentServlet.java）

|  |
| --- |
| package com.demo.servlet;  import com.demo.util.Util;  import com.demo.service.StudentService;  import com.demo.service.impl.StudentServiceImpl;  import com.demo.vo.Student;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  import java.io.IOException;  import java.io.Serializable;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* Student模块的Servlet控制层，负责接收页面传过来的请求参数，根据action参数的值来确定页面要执行的具体操作<br>  \* 而后再调用StudentService业务层的方法来处理具体的业务，最后将处理完成的结果返回或跳转至相应页面  \*/  //@WebServlet("/StudentServlet")  public class StudentServlet extends HttpServlet {  /\*\*  \* 处理Post请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //过滤编码  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  String action = Util.decode(request, "action");  if ("add".equals(action)) {//增加  Student vo = new Student();  //取出页面传进来的各个数据，并设置到Student对象的属性里  String studentName = Util.decode(request, "studentName");  if (studentName != null && !studentName.equals("")) {  vo.setStudentName(studentName);  }  String studentSex = Util.decode(request, "studentSex");  if (studentSex != null && !studentSex.equals("")) {  vo.setStudentSex(studentSex);  }  String studentNo = Util.decode(request, "studentNo");  if (studentNo != null && !studentNo.equals("")) {  vo.setStudentNo(studentNo);  }  String studentBuilding = Util.decode(request, "studentBuilding");  if (studentBuilding != null && !studentBuilding.equals("")) {  vo.setStudentBuilding(studentBuilding);  }  String studentDormitory = Util.decode(request, "studentDormitory");  if (studentDormitory != null && !studentDormitory.equals("")) {  vo.setStudentDormitory(studentDormitory);  }  String studentPhone = Util.decode(request, "studentPhone");  if (studentPhone != null && !studentPhone.equals("")) {  vo.setStudentPhone(studentPhone);  }  String studentText = Util.decode(request, "studentText");  if (studentText != null && !studentText.equals("")) {  vo.setStudentText(studentText);  }  StudentService studentService = new StudentServiceImpl();  //调用Service层增加方法（add），增加记录  studentService.add(vo);  this.redirectList(request, response);  } else if ("delete".equals(action)) {//删除  //取出表要删除的学生记录的主键  long id = Long.parseLong(Util.decode(request, "id"));  StudentService studentService = new StudentServiceImpl();  //调用Service层删除方法（delete），将对应的记录删除  studentService.delete(id);  this.redirectList(request, response);  } else if ("edit".equals(action)) {//修改  //取出页面传进来的各个数据，并设置到Student对象的属性里  Student vo = new Student();  String id = Util.decode(request, "id");  if (id != null && !id.equals("")) {  vo.setId(Long.valueOf(id));  }  String studentName = Util.decode(request, "studentName");  if (studentName != null && !studentName.equals("")) {  vo.setStudentName(studentName);  }  String studentSex = Util.decode(request, "studentSex");  if (studentSex != null && !studentSex.equals("")) {  vo.setStudentSex(studentSex);  }  String studentNo = Util.decode(request, "studentNo");  if (studentNo != null && !studentNo.equals("")) {  vo.setStudentNo(studentNo);  }  String studentBuilding = Util.decode(request, "studentBuilding");  if (studentBuilding != null && !studentBuilding.equals("")) {  vo.setStudentBuilding(studentBuilding);  }  String studentDormitory = Util.decode(request, "studentDormitory");  if (studentDormitory != null && !studentDormitory.equals("")) {  vo.setStudentDormitory(studentDormitory);  }  String studentPhone = Util.decode(request, "studentPhone");  if (studentPhone != null && !studentPhone.equals("")) {  vo.setStudentPhone(studentPhone);  }  String studentText = Util.decode(request, "studentText");  if (studentText != null && !studentText.equals("")) {  vo.setStudentText(studentText);  }  StudentService studentService = new StudentServiceImpl();  //调用Service层更新方法（update），更新记录  studentService.update(vo);  this.redirectList(request, response);  } else if ("get".equalsIgnoreCase(action) || "editPre".equalsIgnoreCase(action)) {//根据主键ID，查询详情信息并跳转到详情页面或编辑页面  Serializable id = Util.decode(request, "id");//取出页面传入的主键，用于查询详情  StudentService studentService = new StudentServiceImpl();  Student vo = studentService.get(id);  request.getSession().setAttribute("vo", vo);  String to = "get".equalsIgnoreCase(action) ? "info" : "edit";//判断是去详情显示页面还是编辑页面  response.sendRedirect("student\_" + to + ".jsp");  } else {//默认去列表页面  this.redirectList(request, response);  }  }  /\*\*  \* 处理Get请求  \*  \* @param request  \* @param response  \* @throws ServletException  \* @throws IOException  \*/  @Override  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doPost(request, response);//Get请求和Post请求的处理是一样的，所以把request、response转交给Post方法就好  }  /\*\*  \* 根据参数，查询出条例条件的记录集合，最后将数据返回给调用处或者将数据集合设置到session域里，再跳转到对应的列表页面  \*  \* @param request  \* @param response  \*/  private void redirectList(HttpServletRequest request, HttpServletResponse response) throws IOException {  //查询列和关键字  String searchColumn = Util.decode(request, "searchColumn");  String keyword = Util.decode(request, "keyword");  Map<String, Object> params = new HashMap();//用来保存控制层传进来的参数(查询条件)  params.put("searchColumn", searchColumn);//要查询的列  params.put("keyword", keyword);//查询的关键字  StudentService studentService = new StudentServiceImpl();  Map<String, Object> map = studentService.list(params);  request.getSession().setAttribute("list", map.get("list"));  Integer totalRecord = (Integer) map.get("totalCount");//根据查询条件取出对应的总记录数，用于分页  String pageNum = Util.decode(request, "pageNum");//封装分页参数  com.demo.util.PageBean<Object> pb = new com.demo.util.PageBean(Integer.valueOf(pageNum != null ? pageNum : "1"), totalRecord);  params.put("startIndex", pb.getStartIndex());  params.put("pageSize", pb.getPageSize());  List list = (List) studentService.list(params).get("list");//根据分页参数startIndex、pageSize查询出来的最终结果list  pb.setServlet("StudentServlet");  pb.setSearchColumn(searchColumn);  pb.setKeyword(keyword);  pb.setList(list);  request.getSession().setAttribute("pageBean", pb);  request.getSession().setAttribute("list", pb.getList());  response.sendRedirect("student\_list.jsp");  }  } |

JavaBean实体模型层定义（Student.java）

|  |
| --- |
| package com.demo.vo;  import java.io.Serializable;  /\*\*  \* 学生（t\_student表对应的Java实体类）  \*/  public class Student implements Serializable {  private Long id;//主键  private String studentName;//姓名  private String studentSex;//性别:男/女  private String studentNo;//学号  private String studentBuilding;//楼栋  private String studentDormitory;//宿舍  private String studentPhone;//电话  private String studentText;//备注  public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public String getStudentName() {  return studentName;  }  public void setStudentName(String studentName) {  this.studentName = studentName;  }  public String getStudentSex() {  return studentSex;  }  public void setStudentSex(String studentSex) {  this.studentSex = studentSex;  }  public String getStudentNo() {  return studentNo;  }  public void setStudentNo(String studentNo) {  this.studentNo = studentNo;  }  public String getStudentBuilding() {  return studentBuilding;  }  public void setStudentBuilding(String studentBuilding) {  this.studentBuilding = studentBuilding;  }  public String getStudentDormitory() {  return studentDormitory;  }  public void setStudentDormitory(String studentDormitory) {  this.studentDormitory = studentDormitory;  }  public String getStudentPhone() {  return studentPhone;  }  public void setStudentPhone(String studentPhone) {  this.studentPhone = studentPhone;  }  public String getStudentText() {  return studentText;  }  public void setStudentText(String studentText) {  this.studentText = studentText;  }  } |

增加信息页面（student\_add.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>添加学生</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">添加学生</a>  <br>  <br>  </div>  <br>  <form action="StudentServlet?action=add" method="post" onsubmit="return check()">  <table class="index-content-table-add">  <tr>  <td width="12%">姓名：</td><td><input class="index-content-table-td-add" type="text" id="studentName" name="studentName" value=""/></td>  </tr>  <tr>  <td width="12%">性别：</td>  <td>  <input name="studentSex" type="radio" value="男" checked="checked"/>&nbsp;&nbsp;&nbsp;男&nbsp;&nbsp;&nbsp;&nbsp;  <input name="studentSex" type="radio" value="女"/>&nbsp;&nbsp;&nbsp;女&nbsp;&nbsp;&nbsp;&nbsp;  </td>  </tr>  <tr>  <td width="12%">学号：</td><td><input class="index-content-table-td-add" type="text" id="studentNo" name="studentNo" value=""/></td>  </tr>  <tr>  <td width="12%">楼栋：</td><td><input class="index-content-table-td-add" type="text" id="studentBuilding" name="studentBuilding" value=""/></td>  </tr>  <tr>  <td width="12%">宿舍：</td><td><input class="index-content-table-td-add" type="text" id="studentDormitory" name="studentDormitory" value=""/></td>  </tr>  <tr>  <td width="12%">电话：</td><td><input class="index-content-table-td-add" type="text" id="studentPhone" name="studentPhone" value=""/></td>  </tr>  <tr>  <td width="12%">备注：</td><td><textarea id="studentText" name="studentText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......"></textarea></td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("studentName").value.trim().length == 0) {  alert("姓名不能为空!");  return false;  }  if (document.getElementById("studentNo").value.trim().length == 0) {  alert("学号不能为空!");  return false;  }  if (document.getElementById("studentBuilding").value.trim().length == 0) {  alert("楼栋不能为空!");  return false;  }  if (document.getElementById("studentDormitory").value.trim().length == 0) {  alert("宿舍不能为空!");  return false;  }  if (document.getElementById("studentPhone").value.trim().length == 0) {  alert("电话不能为空!");  return false;  }  return true;  }  </script>  </html> |

更新信息页面（student\_edit.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>修改学生</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">修改学生</a>  <br>  <br>  </div>  <br>  <form action="StudentServlet?action=edit" method="post" onsubmit="return check()">  <input type="hidden" id="id" name="id" value="${vo.id}"/>    <table class="index-content-table-add">  <tr>  <td width="12%">姓名：</td><td><input class="index-content-table-td-add" type="text" id="studentName" name="studentName" value="${vo.studentName}"/></td>  </tr>  <tr>  <td width="12%">性别：</td>  <td>  <input name="studentSex" type="radio" value="男" ${vo.studentSex=='男'?'checked':''}/>&nbsp;&nbsp;&nbsp;男&nbsp;&nbsp;&nbsp;&nbsp;  <input name="studentSex" type="radio" value="女" ${vo.studentSex=='女'?'checked':''}/>&nbsp;&nbsp;&nbsp;女&nbsp;&nbsp;&nbsp;&nbsp;  </td>  </tr>  <tr>  <td width="12%">学号：</td><td><input class="index-content-table-td-add" type="text" id="studentNo" name="studentNo" value="${vo.studentNo}"/></td>  </tr>  <tr>  <td width="12%">楼栋：</td><td><input class="index-content-table-td-add" type="text" id="studentBuilding" name="studentBuilding" value="${vo.studentBuilding}"/></td>  </tr>  <tr>  <td width="12%">宿舍：</td><td><input class="index-content-table-td-add" type="text" id="studentDormitory" name="studentDormitory" value="${vo.studentDormitory}"/></td>  </tr>  <tr>  <td width="12%">电话：</td><td><input class="index-content-table-td-add" type="text" id="studentPhone" name="studentPhone" value="${vo.studentPhone}"/></td>  </tr>  <tr>  <td width="12%">备注：</td><td><textarea id="studentText" name="studentText" style="width: 60%; height: 100px;padding: 0px 17px;" placeholder="请输入内容......">${vo.studentText}</textarea></td>  </tr>  </table>  <br>  <br>  <br>  &nbsp;&nbsp;&nbsp;<button type="submit" class="btn btn-default btn-sm">提交</button>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">取消</button>  </form>  </div>  </body>  <script type="text/javascript">  //提交之前进行检查，如果return false，则不允许提交  function check() {  //根据ID获取值  if (document.getElementById("studentName").value.trim().length == 0) {  alert("姓名不能为空!");  return false;  }  if (document.getElementById("studentNo").value.trim().length == 0) {  alert("学号不能为空!");  return false;  }  if (document.getElementById("studentBuilding").value.trim().length == 0) {  alert("楼栋不能为空!");  return false;  }  if (document.getElementById("studentDormitory").value.trim().length == 0) {  alert("宿舍不能为空!");  return false;  }  if (document.getElementById("studentPhone").value.trim().length == 0) {  alert("电话不能为空!");  return false;  }  return true;  }  </script>  </html> |

查看详情页面（student\_info.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>学生详情</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">学生详情</a>  <br>  <br>  </div>  <br>  <form>  <table class="index-content-table-add" style="font-size: 18px;">  <tr>  <td>姓名：<b>${vo.studentName}</b></td>  </tr>  <tr>  <td>性别：  <b>${vo.studentSex}</b>  </td>  </tr>  <tr>  <td>学号：<b>${vo.studentNo}</b></td>  </tr>  <tr>  <td>楼栋：<b>${vo.studentBuilding}</b></td>  </tr>  <tr>  <td>宿舍：<b>${vo.studentDormitory}</b></td>  </tr>  <tr>  <td>电话：<b>${vo.studentPhone}</b></td>  </tr>  <tr>  <td>备注：<b>${vo.studentText}</b></td>  </tr>  </table>  <br>  <button type="button" class="btn btn-success btn-sm" onclick="javascript:history.back(-1);">返回</button>  </form>  </div>  </body>  </html> |

列表与查询页面（student\_list.jsp）

|  |
| --- |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8"/>  <title>学生管理</title>  <link rel="stylesheet" type="text/css" href="css/index.css"/>  </head>  <body>  <jsp:include page="menu.jsp"/>  <div class="index-content">  <div class="index-content-operation">  <a class="info-detail">学生管理</a>  <br>  <br>  </div>  <br>  <div class="index-content-operation">  <button class="btn btn-default btn-sm" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="window.location.href='student\_add.jsp'">添加</button>  <div class="index-content-operation-search"><input id="search\_keyword" placeholder="姓名" type="text" name="search\_keyword"/><input type="hidden" id="searchColumn" name="searchColumn" value="student\_name"/><button class="btn btn-primary btn-sm" onclick="searchList()">搜索</button></div>  </div>  <br>  <table class="table table-striped table-hover table-bordered">  <thead>  <tr class="index-content-table-th">  <th>姓名</th>  <th>性别</th>  <th>学号</th>  <th>楼栋</th>  <th>宿舍</th>  <th>电话</th>  <th>操作</th>  </tr>  </thead>  <tbody>  <c:forEach items="${list}" var="vo">  <tr class="index-content-table-td">  <td>${vo.studentName}</td>  <td>${vo.studentSex}</td>  <td>${vo.studentNo}</td>  <td>${vo.studentBuilding}</td>  <td>${vo.studentDormitory}</td>  <td>${vo.studentPhone}</td>  <td>  <button class="btn btn-success btn-sm" style="padding: 0px 1px;" onclick="window.location.href='StudentServlet?action=get&id=${vo.id}'">详情</button>&nbsp;  <button class="btn btn-info btn-sm" style="padding: 0px 1px;"  <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if>  onclick="window.location.href='StudentServlet?action=editPre&id=${vo.id}'">编辑</button>&nbsp;  <button class="btn btn-danger btn-sm" style="padding: 0px 1px;" <c:if test="${loginUser.userType != '管理员'}">disabled="disabled" title="没有权限！！！"</c:if> onclick="if(window.confirm('将要删除：${vo.studentName}？'))window.location.href='StudentServlet?action=delete&id=${vo.id}'">删除</button>  </td>  </tr>  </c:forEach>  </tbody>  </table>  <div style="float: right;padding-right: 10px;color: #515151;"><jsp:include page="split.jsp"/></div>  </div>  </body>  <script>  function searchList() {  window.location.href = "StudentServlet?action=list&searchColumn="+document.getElementById("searchColumn").value+"&keyword=" + document.getElementById("search\_keyword").value;  }  </script>  </html> |

# 9 总结展望

学生宿舍管理系统的基本功能已经实现。虽然时间很短暂，但却是对几年学习中理论知识和实践相结合的一次综合检验。通过这次项目设计，收到了比以往理论课程还要大的收益，虽然由于时间仓促及本人的能力有限，系统还有很多不尽人意的地方：比如说界面不够美观；有些功能还不够完善和强大；代码的重用性不够高；一些细节的问题还没有解决。这些都需要平时经验的积累和对技术的熟练掌握，希望在以后的工作学习中能有进一步的提高。

致谢

在系统开发的一个多星期里，我受到了指导老师细心指导，老师严谨的指导态度与深厚的理论知识都让我受益非浅，从他身上我学到了很多的东西，无论是理论还是实践都使我的知识有很大的提高。谢谢老师为本系统提供的帮助，让我能够顺利地完成本系统的开发。也感谢那些对我提供帮助的同学，没有他们就不会有现在的系统.

本文引用了数位学者的研究文献，如果没有各位学者的研究成果的帮助和启发，我将很难完成本篇论文的写作。感谢我的同学和朋友，在我写论文的过程中给予我了很多素材，还在论文的撰写和排版的过程中提供热情的帮助。

展望

由于本人现有水平和经验有限，系统在设计过程中，难免存在不足之处，使得本系统方案设计仍存在许多有待完善和改进的地方，比如说，在程序结构上系统设计得还不够紧密，对一些问题，如数据库的设计上没有统一，造成部分数据冗余。其次，安全性上考虑不够仔细等等。比如说，没有考虑到跨平台可能面临的一些并发性问题、数据共享问题，多个用户会同时访问主页，会造成服务器端网络的阻塞，致使系统性能的下降。比如说，数据的备份与恢复。数据库的备份与恢复原状一个十分重要的功能，该部分功能的好坏，直接影响到数据的安全及系统的完整性和易用性，没有数据备份与恢复功能的系统不是一个完整的系统。数据不能够及时备份，出现问题时不能完全恢复，这样的数据库应用软件，其使用价值将大打折扣，易用性也无从谈起。此外，还有数据库的安全性和保密性。要对数据进行加密来防止数据库被非法修改和破坏。但由于时间关系和技术有限，本系统的这些功能都未得以实现，所以系统仍然需要进一步完善。总的说来，本系统还不成熟，在某些细节方面可能还没有考虑到，功能不是很强大，还不能提供决策性的分析功能，对业务流程的控制不是很灵活。但对它的改善和功能的完备并不会随着毕业设计的结束而终止。后续的工作还需付出更大的努力去完善，本人将在原来的基础上，着眼未来，追踪计算机新技术发展，并应用到系统持续开发中，不断刻苦钻研，不断提高开发水平，对系统进一步优化。

# 10 参考文献

[1] 李洋. SSM框架在Web应用开发中的设计与实现[J]. 计算机技术与发展, 2021, 26(12):190-194.

[2] Walls C, Breidenbach R. Spring in action[M]. Manning Publications Co. 2020.

[3] 赵小涛. 基于SSM框架的铁路技术规章管理系统的设计与实现[D]. 北京: 北京交通大学, 2021.

[4] 杨阳, 汤光恒. 基于Spring Boot的高校部门测评系统的设计与实现[J]. 福建电脑, 2020, 34(8):128-129.

[5] 徐雯, 高建华. 基于Spring MVC及MyBatis的Web应用框架研究[J]. 微型电脑应用, 2021, 28(7):1-4, 10.

[6] 武强. 基于B/S三层架构下的基层连队管理信息系统的设计与实现[D]. 长春: 长春工业大学, 2021.

[7] 姜博文 . 基于B/S 架构的资产数据管理系统的设计与实现[D]. 北京: 北京交通大学, 2020.

[8] 王书爱 . 面向对象程序设计的应用[J]. 电脑知识与技术,2021, 7(29):7289-7290, 7299.

[9] 程春蕊, 刘万军. 高内聚低耦合软件架构的构建[J]. 计算机系统应用, 2020, 18(7):19-22.

[10] 盛舒婷.基于Android的课程设计管理系统的设计与实现[D],2021

[11] 李恺.基于Web的课程设计题目管理系统的设计与实现[D],2020