**中北大学软件学院**

实训说明书

实训名称:  *云计算与大数据*

题目名称： 实验室预约管理系统

专 业： 软件工程

班 级： 14140Y01

小组成员

学号： 1414010513 姓名：张丹 成绩：

学号： 1414010231 姓名：王鹏 成绩：

学号： 1414010507 姓名：郑敏 成绩：

学号： 1414010609 姓名：崔佩佩 成绩：

指导教师： 李华玲

2016 年 5 月

中 北 大 学

实训任务书

|  |  |  |  |
| --- | --- | --- | --- |
| **实 训 名 称：** | 面向对象程序设计(JAVA)实训 | | |
| **学 院、系：** | 软件学院 | | |
| **专 业：** | 软件工程 | | |
| **班 级：** | 14140Y01班 | | |
| **学 生 姓 名：** | 王鹏 | **学 号：** | 1414010231 |
|  | 郑敏 | **学 号：** | 1414010507 |
|  | 张丹 | **学 号：** | 1414010513 |
|  | 崔佩佩 | **学 号：** | 1414010609 |
| **设 计 题 目:** | 实验室预约管理系统 | | |
| **起 迄 日 期:** | 2016年5月9日~2016年5月27日 | | |
| **指 导 教 师:** | 李华玲 | | |

发任务书日期: 2016年5月9日**实 训 任 务 书**

|  |
| --- |
| 1．实训的目的和要求： |
| 通过实训，使学生进一步理解和巩固面向对象程序设计的基本概念、思想和核心技术；掌握面向对象程序设计的知识和技能；熟悉使用面向对象技术进行软件开发的过程；加强学生对Java知识的全面掌握；提高分析问题、解决问题的能力；锻炼动手能力、创新能力和综合应用的能力。  **基本要求：**  （1）使用Java或Java web，数据库自由选择（MySql、SQLServer或ORACLE）；  （2）按照软件工程的思想完成系统开发过程（包括需求分析、系统设计、编码实现和测试）；  （3）最终的软件系统要求数据充实、界面友好、使用方便。 |
| 2．实训的具体工作内容及分工： |
| **设计内容：**  现在高校提倡实验室向学生免费开放，使实验室的资源充分利用。为此学生需要根据自己的时间提前进行实验室预约，请开发一个面向全校学生的实验室预约管理系统。  **主要功能及要求：**  （1）全校实验室信息的管理（录入、修改、删除等基本操作）；  （2）学生使用学号和密码登录后能实现实验室的预约，并显示自己的预约信息；  （3）实验室管理员登录系统后能查看自己所负责实验室当天或一周的预约情况； |

**实 训 任 务 书**

|  |  |
| --- | --- |
| 3．对实训成果的要求： | |
| 提交实训说明书和光盘（每班统一制作1张光盘，每组一个目录）。  每组的光盘目录内含：  （1）源程序文件；  （2）数据库备份文件；  （3）实训说明书文档（另需交打印文档一份）。 | |
| 4．实训工作进度计划： | |
| 起 迄 日 期 | 工 作 内 容 |
| **2016年**  5月9日 **~** 5月11日  5月12日 **~** 5月24日  5月25日 **~** 5月26日  5月27日 | 选择实训题目，明确实训任务，完成需求分析及系统分析与设计；  功能模块代码实现、调试，完善实例数据，系统测试；  撰写实训说明书文档；  验收程序、答辩。 |

任务分工情况说明

|  |  |  |
| --- | --- | --- |
|  | 姓名 | 分工 |
| 组长 | 张丹 | （1）全校实验室信息的管理（录入、修改、删除等基本操作）；  （2）当天或一周的预约人数情况查询  （3）实训说明书（概述、需求分析、总体设计） |
| 组员 | 郑敏 | 1. 实验室预约管理系统首页   （2）总管理员查询实验室一周预约情况  （3）实训说明书（详细设计） |
| 组员 | 王鹏 | 1. 分页显示实验室一天预约情况 2. 信息预约试验时自动获取当天日期   （3）实训说明书（测试、心得体会、参考文献） |
| 组员 | 崔佩佩 | 1. 数据库的设计（全校实验室信息表、学生表） 2. 预约实验室信息的管理（录入、修改、删除等基本操作）； 3. 全校总管人、每位学生、实验室管理员的登录   （4）实训说明书（数据库设计、维护） |

**目录**

**1 引言………**

**2 总体设计………**

**3 详细设计………**

**4 系统实现………**

**5 心得体会………**

**6 参考文献………**

**第1章 引言**

1.1编写目的

实验室预约管理系统是学校为了更好的在预约实验室的过程中对学生进行管理，让学生和管理人员及时的获得最新的实验室信息的系统。管理人员对实验室的有关信息的更改能够及时发布出来，学生可以方便的选择所要实训的地点，同时管理员可以查询学校学生的基本信息，及时向管理员提供学生信息的系统。该系统由管理员设计功能完善的数据库，有效、安全的保存了相关数据，使管理系统更加严谨、安全、方便。整个系统对于管理员和学生来说操作简便，并且系统设计的界面友好、灵活、实用、安全，有效帮助管理员完成了学生与实验室之间的信息管理的全过程。该预约管理系统利用计算机的信息系统和各种软件资源，为在校学生提供了一个完整的实验室提供的项目开发过程，同时也是学生将所学知识与具体项目实践相结合的环境和平台！

**1.3用户类型**

1）学生： 能较熟练的操作计算机，具有一定的编程能力，能查询实验室的信息进行选择实验室，同样可以退出实验室选择其他实验室。可以对学生的信息进行查询，并可以修改自己的基本信息及密码；

2）管理员：可以熟练操作计算机，拥有软件知识基础并有较强的数据库管理能力对学生和实验室之间的预约关系进行管理和统计。可以按条件查询实验室的信息并且对其进行增加、修改、删除操作，可以按条件查询学生的基本信息并对其进行增加、修改、删除操作，管理员还可以修改自己登录的密码。

**第2章 总体设计**

## 2.1 总体功能要求

（1）开发者应在充分分析需求的基础上，选择采用B/S结构或者C/S结构。

（2）实验室预约管理系统系统应依照《软件工程导论》进行设计和实现。

（3）开发者根据项目具体特点选择采用面向对象的方法来进行分析、设计和开发。

## 2.2 软件开发平台要求

目前软件平台为：

开发环境：Eclipse

服务器系统：Tomcat

数据库管理系统：MySql

**2.3 软件项目的开发实施过程管理要求**

### 2.3.1 软件项目实施过程总体要求

               （一）小组组长拿手写或打印出来的需求分析与设计文档和分工情况表找指导老师当面确认。指导老师对其进行评审，并提出整改意见。

    （二）通过评审后，小组成员根据整改意见完善说明书，经过指导老师认可后进行软件开发。系统开发工作按照需求分析、概要设计、详细设计、编码、测试等几个阶段进行，在开发过程中，开发者需分阶段提交相关文档。

    （三）在软件开发工作完成后，开发者应向指导老师提交完整的软件文档，指导老师对所做系统进行验收审查。

### 2.3.2 软件项目实施变更要求

在系统开发过程中，需求或设计不可避免地需要发生变更，相关变更必须经过指导老师同意方可进行。在需求或设计发生变更时，需要对原有文档进行修改，并提供完整的变更记录，  以使变更处于可控制的状态。变更单如下表所示：

表2.1 变更单

|  |  |
| --- | --- |
| 需求变更申请 | |
| 申请变更的需求文档 | 输入名称，版本，日期等信息 |
| 变更的内客及其理由 |  |
| 评估需求变更将对  项目造成的影响 |  |
| 申请人签字 |  |
| 变更申请的审批意见 | |
| 指导老师签字 | 审批意见：                               签字   日期 |
| 更改需求文档 | |
| 变更后的  需求文档 | 输入名称，版本，完成日期等信息 |
| 更改人签字 |  |
| 重新评审需求文档 | |
| 需求评审小组签字 | 评审意见：                               签字   日期 |
| 变更结束 | |

### 

### 2.3.3 软件项目实施里程碑控制

系统开发小组将分四个阶段进行。

               （一） 概述+需求分析；

               （二） 详细设计+数据库设计；

               （三） 编码+测试；

              （四） 运行+维护。

# 

# 2.4. 软件开发

经过指导老师认可后进行软件开发。系统开发必须严格按照软件工程的要求进行。开发过程包括开发者的活动和任务。此过程由软件需求分析、概要设计、详细设计、编码、测试、验收、鉴定等活动组成。

## 2.4.1软件的需求分析

### 2.4.1.1需求分析

首先，开发者和指导老师应共同对实验室管理系统的应用需求作充分的调研，提交完整的需求分析报告。在需求分析报告中必须描述的基本问题是：功能、性能、强加于实现的设计限制。应当避免把设计或项目需求写入需求分析报告中。它必须说明由软件获得的结果，而不是获得这些结果的手段。

软件需求可以用若干种方法来表达，如通过输入、输出说明；使用代表性的例子；用规范化的模型。开发者应尽可能地使用模型的方式，因为这是表达复杂需求的精确和有效的方法。比如用统一建模语言（UML）来描述需求。

编写需求分析报告的要求

a．无歧义性

对最终系统的每一个特性用某一术语描述；若某一术语在某一特殊的行文中使用时具有多种含义，那么应对该术语的每种含义做出解释并指出其适用场合。

b．完整性

需求分析报告应该包括全部有意义的需求，无论是关系到功能的、性能的、设计约束的、还是关系到外部接口方面的需求；对所有可能出现的输入数据的响应予以定义，要对合法和非合法的输入值的响应做出规定。

c．可验证性

需求分析报告描述的每一个需求应是可以验证的。可以通过一个有限处理过程来检查系统是否满足需求。

d．一致性

在需求分析报告中的各个需求的描述不能互相矛盾。

e．可修改性

需求分析报告应具有一个有条不紊、易于使用的内容组织；没有冗余，即同一需求不能在需求分析报告中出现多次。

f．可追踪性

每一个需求的源流必须清晰，在进一步产生和改变文件编制时，可以方便地引证每一个需求。

g．运行和维护阶段的可使用性

需求分析报告必须满足运行和维护阶段的需要。在需求分析报告要写明功能的来源和目的。

### 

### 2.4.1.2需求分析报告的编制者

需求分析报告应由开发者完成。系统开发者根据老师提出的性能需求，结合软件开发编写需求分析。

### 2.4.1.3 需求报告评审

在软件需求分析工作完成后，系统开发者应向交通局提交《软件需求分析报告》。指导老师对需求进行评审，以决定软件需求是否完善和恰当。评审完成后，就可以进入软件的设计阶段。

2.4.1.4需求分析内容

（1）全校实验室信息的管理（录入、修改、删除等基本操作）；

（2）学生使用学号和密码登录后能实现实验室的预约，并显示自己的预约信息；

（3）实验室管理员登录系统后能查看自己所负责实验室当天或一周的预约情况；

## 2.5 软件的概要设计

### 2.5.1 概要设计

在指导老师和开发小组双方认可的《需求分析报告》基础上，开发小组进行下—步的工作。首先，开发者需要对软件系统进行概要设计，即系统设计。概要设计需要对软件系统的设计进行考虑，包括系统的基本处理流程、系统的组织结构、模块划分、功能分配、接口设计、运行设计、数据结构设计和出错处理设计等，为软件的详细设计提供基础。

### 2.5.2 编写概要设计的要求

a．一致性

概要设计的要求应该与需求分析报告所描述的需求一致。同时，概要设计的各项要求之间也应该一致。

b．合理性

概要设计所提出的设计方法和标准应该是合理的、恰当的。

c．可追踪性

对概要设计所提出的各项要求应该可以得到它的清晰的源流，即在需求分析报告客户有明确的需求描述。

d．可行性

根据概要设计进行详细设计、操作和维护应该是可行的。

### 2.5.3 概要设计报告的编写者

概要设计报告由开发小组根据需求分析报告的要求进行编写。

### 2.5.4 概要设计和需求分析、详细设计之间的关系和区别

 需求分析不涉及具体的技术实现，而概要设计注重于从宏观上和框架上来描述采用何种技术手段、方法来实现这些需求。详细设计相对概要设计更注重于微观上和框架内的设计， 是编码的依据。概要设计是指导详细设计的依据。

### 2.5.5 概要设计的评审

在软件概要设计工作完成后，软件开发小组应向指导老师提交《软件系统概要设计报告》。在指导老师对《概要设计报告》评审通过后，即可进入详细设计阶段。

2.5.6 概要设计内容

1. 总管理员登录——全校实验室信息表——增删改查
2. 学生登录——预约实验室——显示预约信息
3. 实验室管理员登录——查看自己负责的实验室的当天或一周预约情况

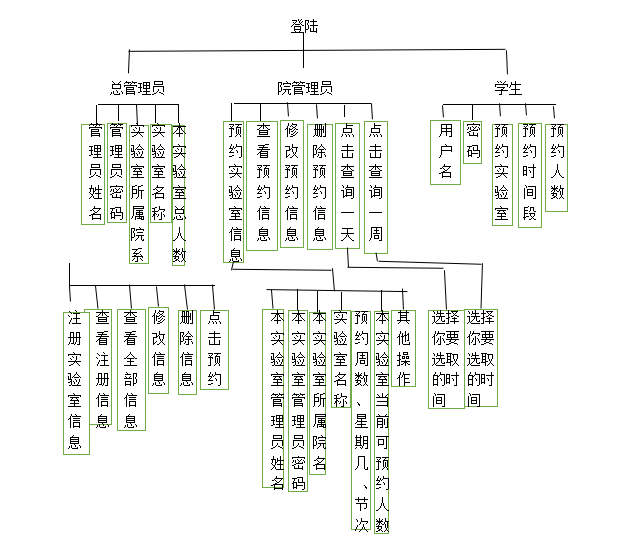


图2.1 系统结构图

2.6 数据库设计

学生，院管理员，总管理员登录表

表2-1（登录）

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | login |  |  |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| username | varchar | 非空 | 主码 |
| password | varchar | 非空 |  |
| type | varchar | 非空 |  |

表2-2（学生预约）

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | student |  |  |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| bookusername | varchar | 非空 | 主码 |
| password | varchar | 非空 |  |
| dept | varchar | 非空 |  |
| name | varchar | 非空 | 外码(参照表manager) |
| week | varchar | 非空 |  |
| day | varchar | 非空 |  |
| time | varchar | 非空 |  |
| sbook | varchar | 非空 |  |

表2-3（总管理员表）

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | manager |  |  |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| username | varchar | 非空 |  |
| password | varchar | 非空 |  |
| dept | varchar | 非空 |  |
| name | varchar | 非空 | 主码 |
| scount | varchar | 非空 |  |

表2-4（院管理员表）

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | deptmanager |  |  |
| 列名 | 数据类型 | 空/非空 | 约束条件 |
| username | varchar | 非空 | 外码(参照student表) |
| password | varchar | 非空 |  |
| dept | varchar | 非空 |  |
| name | varchar | 非空 | 主码 |
| scount | varchar | 非空 |  |

**2.7 设计页面**



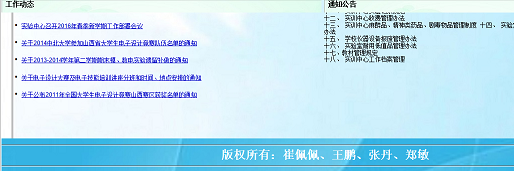


图1



图2



图3



图4



图5



图6



图7

**第3章 详细设计**

**3.1 外部接口**

本系统暂无外部接口

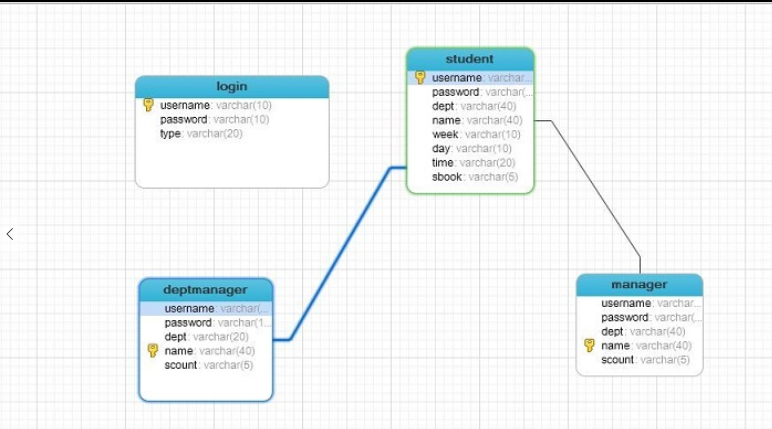
**3.2 运行设计**

此详细设计详细描述管理员以及学生两个模块部分。

E-R图是用来描述现实世界的概念模型，构成E-R图的基本要素是实体型、属性和联系，其表示方法为：实体型(Entity)：具有相同属性的实体具有相同的特征和性质，用实体名及其属性名集合来抽象和刻画同类实体;在E-R图中用矩形表示，矩形框内写明实体名；比如学生张三等。属性(Attribute)：实体所具有的某一特性，一个实体可由若干个属性来刻画。在E-R图中用椭圆形表示，并用无向边将其与相应的实体连接起来；比如学生的姓名、学号、性别、都是属性。如果是多值属性的话，再椭圆形外面再套实线椭圆。如果是派生属性则用虚线椭圆表示。联系(Relationship)：联系也称关系，信息世界中反映实体内部或实体之间的联系。实体内部的联系通常是指组成实体的各属性之间的联系；实体之间的联系通常是指不同实体集之间的联系。在E-R图中用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型（1 : 1，1 : n或m : n）。

该系统的模型主要描述了学生，管理员的属性，还有他们之间的关系。

**数据库关系E-R图：**

** 图3-1**

**第4章 系统实现**

部分功能实现：

登录方式：学生登录

院管理员登录

总管理员登录

学生，院管理员，总管理员的用户名，密码已给定，不同身份登录显示不同的信息。

学生登录：学生通过已定的用户名和密码实现登录，进行实验室预约，学生可通过用户名，密码选择实验室所在学院，实验室名称，预约周次，预约周几，预约时间段。

预约成功后，学生可以查看预约信息，也可修改预约或者取消预约。

第二次登陆后，直接选择“已预约”，输入学号后，可查个人预约信息。

1. **心得体会**

1》通过实训，让我们进一步理解和巩固面向对象程序设计的基本概念、思想和核心技术；掌握面向对象程序设计的知识和技能；熟悉使用面向对象技术进行软件开发的过程；加强学生对Java知识的全面掌握；提高分析问题、解决问题的能力；锻炼动手能力、创新能力和综合应用的能力。

2》通过这次课程设计练习，让我体会到了一个团队的团队精神的重要性，也知道个人的能力知识占一部分，但那也是必不可少的经验。同时，我们在遇到不懂的问题的时候要学会查询有关书籍，扩充自己的知识面。并且要虚心向他人和老师请教，只有这样才可以不断提高自己，充实自己。只要每个人都付出了就一定会有收获的。而不是在接除困难的时候退缩，也就是说只要我们全力以赴，用心去做每一件事，就一定会取得一定的成就的。这次的课程设计中的不足，让我找到了奋斗的动力。因此，在以后的学习中我一定会更加珍惜这来之不易的学习机会，努力学习知识，从而为以后的工作做好充分的准备。

3》从刚开始进行需求分析，进行了大量的需求调研，得到了该系统的客观需求。对本系统进行必要的分析和建模，然后对整个系统的功能架构进行了深入的分析，完成了系统的概要设计和详细设计部分。根据以上各个环节的成果和数据库设计方面的知识顺利完成了该系统数据库的设计。最后一个相当重要的环节是系统测试，在这个环节中，根据整个系统的用户需求对各个模块、各个接口、运行环境以及相关数据进行测试。

本系统主要有以下几个优点:

（1）节省时间。管理员和学生可以很方便地在网上进行实验预约，管理员和学生不需要花费很多时间在实验预约方面，这样在很大程度上节省了时间。学生向管理员提出预约申请，只有管理员审核通过以后，该实验才能提交给总管理员，使总管理员对其进行审核。这样，学生会有更多的时间对改实验进行准备和进行实验。

提高学生积极性。学生没有必要像以往一样经过一系列繁琐的过程进行实验预约，这样在很大程度上提高的学生的积极性。学生乐于通过实验获取新的知识，这样可以提高实验教学的效果。

（2）便于管理。管理员可以很方便地在网上对教师和学生的实验进行审核，而不像传统的人工处理方式既花费时间，又容易出错。管理员可以根据教师和学生的教学需求，对实验的时间和地址进行合理的变动。

这个系统虽然在很大程度上提高了实验预约的效率，但是由于时间和技术的限制，该系统依然存在不足。比如，当学生大量的进行实验室预约时，会不会造成数据流的拥挤，会不会对系统的稳定性造成影响。以及在该系统中加入实验室的空闲状态以及个实验室适合那些类型的实验，以便于学生更加有效的选择实验室，这样，可以减少实验管理员为学生安排实验室的工作负担。所以，应该更深一步的进行需求分析，对部分功能进行改进。

**第六章 参考文献**

【1】明日科技编著. Java Web从入门到精通.北京:清华大学出版社. 2016

【2】刘爽英 王丽芳 李欣然 张元编著. 数据库原理及应用.北京:清华大学出版社. 2016

【3】李刚.2007.轻量级J2EE企业应用实战.Struts+Spring+Hibernate 整合开发.北京：电子工业出版社:128-159

【4】曹晓刚.2005.深入浅出Hibernate.北京：电子出版社

【5】程杰.2007.大话设计模式.北京：清华大学出版社:48-49

【6】张海潘.2002.软件工程导论.清华大学出版社

【7】孙卫琴.2006.Java面向对象编程.人民邮电出版社:168-172

【8】匿名.2013.数据库逻辑结构设计.http://book.51cto.com/art/201302/381567.htm

【9】阎宏.2004.我的J2EE成功之路.清华大学出版社

克雷格.2003.系统的软件测试.杨海燕译.电子工业出版社

【10】 李旭成.2010.J2EE实用教程.电子工业出版社出版社 王珊.2006.数据库系统概论.高等教育出版社

【11】孙鑫.2006．Java Web开发详解.电子工业出版社:189-274

【12】林上杰.2004.林康司编著．JSP2.0技术手册.电子工业出版社:3-6

【13】唐四薪.2009.基于Web标准的网页设计与制作.北京:清华大学出版社：14

【14】肖磊.2010.Java实用教程.北京：人民邮电出版社：245

【15】李绪成.2011.Java EE实用教程—基于WebLogic和Eclipse.北京：电子工业出版社：78

【16】林上杰.2004.JSP2.0技术手册.北京:电子工业出版社

【17】蔡剑.2004.Java Web应用开发：J2EE和Tomcat.北京：清华大学出版社

**附录：**

①学生dao:

package nuc.sw.dao;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import nuc.sw.bean.Member;

import nuc.sw.bean.LoginBean;

import nuc.sw.bean.manager;

import nuc.sw.dbc.DBUtil;

public class MemberDao {

//判断是不是合法用户

private Connection conn=null;

private PreparedStatement ps=null;

private ResultSet rs=null;

public boolean isUser(LoginBean lb){

boolean flag=false;

//获取连接

conn=DBUtil.getConn();

//操作数据

String sql="select username,password,type from login where username=? and password=?and type=?";

try {

ps=conn.prepareStatement(sql);

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

ps.setString(1,lb.getUsername());

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

ps.setString(2,lb.getPassword());

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

ps.setString(2,lb.getPassword());

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

rs=ps.executeQuery();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

if(rs.next()){

flag=true;

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return flag;

}

//判断是否预约成功

public boolean doInsert(Member m){

boolean flag=false;

//获取链接

conn=DBUtil.getConn();

//操作数据

try {

String sql="insert into student values(?,?,?,?,?,?,?,?)";

ps=conn.prepareStatement(sql);

ps.setString(1,m.getUsername());

ps.setString(2,m.getPassword());

ps.setString(3,m.getDept());

ps.setString(4,m.getName());

ps.setString(5,m.getWeek());

ps.setString(6,m.getDay());

ps.setString(7,m.getTime());

ps.setString(8,m.getSbook());

if(ps.executeUpdate()>0){

flag=true;

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return flag;

}

//根据用户名查询所有用户的信息

public List<Member> findAllDay(String day){

List<Member> mlistday=new ArrayList<Member>();

Member m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据 查询到的数据放到m中

String sql="select \* from student where day=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,day);

rs=ps.executeQuery();

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

try {

while(rs.next()){

//将查询处理的结果集里的信息封装到m

m=new Member();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setWeek(rs.getString(5));

m.setDay(rs.getString(6));

m.setTime(rs.getString(7));

m.setSbook(rs.getString(8));

mlistday.add(m);

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return mlistday;

}

//根据用户名查询所有用户的信息

public List<Member> findAllWeek(String week){

List<Member> mlist=new ArrayList<Member>();

Member m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据 查询到的数据放到m中

String sql="select \* from student where week=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,week);

rs=ps.executeQuery();

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

try {

while(rs.next()){

//将查询处理的结果集里的信息封装到m

m=new Member();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setWeek(rs.getString(5));

m.setDay(rs.getString(6));

m.setTime(rs.getString(7));

m.setSbook(rs.getString(8));

mlist.add(m);

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return mlist;

}

//显示个人预约信息

public Member findByUsername(String username){

Member m=null;

//获取连接

conn=DBUtil.getConn();

System.out.println(username);

//操作数据

String sql="select \* from student where username=?";

try {

ps=conn.prepareStatement(sql);

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

ps.setString(1,username);

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

rs=ps.executeQuery();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

if(rs.next()){

//将查询到的数据封装到m中

m=new Member();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setWeek(rs.getString(5));

m.setDay(rs.getString(6));

m.setTime(rs.getString(7));

m.setSbook(rs.getString(8));

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return m;

}

public boolean doModify(Member m){

boolean flag=false;

//获取链接

conn=DBUtil.getConn();

//操作数据

String sql="update student set username=?,password=?,dept=?,name=?,week=?,day=?,time=?,sbook=? where username=?";

try {

PreparedStatement ps=conn.prepareStatement(sql);

ps.setString(1,m.getUsername());

ps.setString(2,m.getPassword());

ps.setString(3,m.getDept());

ps.setString(4,m.getName());

ps.setString(5,m.getWeek());

ps.setString(6,m.getDay());

ps.setString(7,m.getTime());

ps.setString(8,m.getSbook());

ps.setString(9,m.getUsername());

if(ps.executeUpdate()>0){

flag=true;

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return flag;

}

//取消预约

public Member doDelete(String username){

Member m=null;

//获取链接

conn=DBUtil.getConn();

//操作数据

String sql="delete from student where username=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,username);

if(ps.executeUpdate()>0){

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return m;

}

public ResultSet Servlet(LoginBean lb){

ResultSet rs=null;

conn=DBUtil.getConn();

try{

String sql="select \* from login where username=? and password=? and type=?";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setString(1, lb.getUsername());

ps.setString(2, lb.getPassword());

ps.setString(3, lb.getType());

rs=ps.executeQuery();

}

catch(Exception e){

e.printStackTrace();

}

return rs;

}

public int getId(int Id)

{

int m=0;

//获取链接

conn=DBUtil.getConn();

//操作数据

try{

String sql="select count(\*) from student Where Id=?";

ps=conn.prepareStatement(sql);

ps.setInt(1, Id);

rs=ps.executeQuery();

if(rs.next()){

rs.getInt(1);

}

}

catch(SQLException ex){

ex.printStackTrace();

}

return m;

}

}

②院管理员dao：

package nuc.sw.dao;

import nuc.sw.dbc.\*;

import nuc.sw.bean.\*;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

//import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import nuc.sw.dbc.\*;

public class ManagerDao {

private Connection conn=null;

private PreparedStatement ps=null;

private ResultSet rs=null;

public boolean doInsert(manager m){

boolean flag=false;

conn=DBUtil.getConn();

String sql="insert into manager values(?,?,?,?,?)";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,m.getUsername());

ps.setString(2,m.getPassword());

ps.setString(3,m.getDept());

ps.setString(4,m.getName());

ps.setString(5,m.getScount());

if( ps.executeUpdate()>0){

flag=true;

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return flag;

}

//根据用户名查询所有用户的信息

public List<manager> findAll(String dept){

List<manager> mlist=new ArrayList<manager>();

manager m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据 查询到的数据放到m中

String sql="select \* from manager where dept=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,dept);

rs=ps.executeQuery();

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

try {

while(rs.next()){

//将查询处理的结果集里的信息封装到m

m=new manager();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setScount(rs.getString(5));

mlist.add(m);

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return mlist;

}

public List<manager> findAll(){

List<manager> mlist=new ArrayList<manager>();

manager m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据 查询到的数据放到m中

String sql="select \* from manager";

try {

ps=conn.prepareStatement(sql);

rs=ps.executeQuery();

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

try {

while(rs.next()){

//将查询处理的结果集里的信息封装到m

m=new manager();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setScount(rs.getString(5));

mlist.add(m);

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return mlist;

}

//根据用户名查询该用户的信息

public manager findByUsername(String username){

manager m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据 查询到的数据放到m中

String sql="select \* from manager where username=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,username);

rs=ps.executeQuery();

if(rs.next()){

//将查询处理的结果集里的信息封装到m

m=new manager();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setScount(rs.getString(5));

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return m;

}

//根据用户名修改该用户的信息

public boolean doUpdate(manager m){

boolean flag2=false;

//获取连接

conn=DBUtil.getConn();

//操作数据

String sql="update manager set username=?,password=?,dept=?,name=?,scount=? where username=?";

try {

ps=conn.prepareStatement(sql);

} catch (SQLException e1) {

// TODO 自动生成的 catch 块

e1.printStackTrace();

}

try {

ps.setString(1,m.getUsername());

ps.setString(2,m.getPassword());

ps.setString(3, m.getDept());

ps.setString(4, m.getName());

ps.setString(5, m.getScount());

ps.setString(6,m.getUsername());

if(ps.executeUpdate()>0){

flag2=true;

}

if(ps!=null)

ps.close();

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return flag2;

}

//根据用户名删除该用户的信息

public manager doDelete(String username){

manager m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据

String sql="delete from manager where username=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,username);

if(ps.executeUpdate()>0){

return m;

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return m;

}

}

③管理员dao

package nuc.sw.dao;

import nuc.sw.dbc.\*;

import nuc.sw.bean.\*;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

//import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import nuc.sw.dbc.\*;

public class ManagerDao {

private Connection conn=null;

private PreparedStatement ps=null;

private ResultSet rs=null;

public boolean doInsert(manager m){

boolean flag=false;

conn=DBUtil.getConn();

String sql="insert into manager values(?,?,?,?,?)";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,m.getUsername());

ps.setString(2,m.getPassword());

ps.setString(3,m.getDept());

ps.setString(4,m.getName());

ps.setString(5,m.getScount());

if( ps.executeUpdate()>0){

flag=true;

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return flag;

}

//根据用户名查询所有用户的信息

public List<manager> findAll(String dept){

List<manager> mlist=new ArrayList<manager>();

manager m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据 查询到的数据放到m中

String sql="select \* from manager where dept=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,dept);

rs=ps.executeQuery();

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

try {

while(rs.next()){

//将查询处理的结果集里的信息封装到m

m=new manager();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setScount(rs.getString(5));

mlist.add(m);

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return mlist;

}

public List<manager> findAll(){

List<manager> mlist=new ArrayList<manager>();

manager m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据 查询到的数据放到m中

String sql="select \* from manager";

try {

ps=conn.prepareStatement(sql);

rs=ps.executeQuery();

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

try {

while(rs.next()){

//将查询处理的结果集里的信息封装到m

m=new manager();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setScount(rs.getString(5));

mlist.add(m);

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return mlist;

}

//根据用户名查询该用户的信息

public manager findByUsername(String username){

manager m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据 查询到的数据放到m中

String sql="select \* from manager where username=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,username);

rs=ps.executeQuery();

if(rs.next()){

//将查询处理的结果集里的信息封装到m

m=new manager();

m.setUsername(rs.getString(1));

m.setPassword(rs.getString(2));

m.setDept(rs.getString(3));

m.setName(rs.getString(4));

m.setScount(rs.getString(5));

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return m;

}

//根据用户名修改该用户的信息

public boolean doUpdate(manager m){

boolean flag2=false;

//获取连接

conn=DBUtil.getConn();

//操作数据

String sql="update manager set username=?,password=?,dept=?,name=?,scount=? where username=?";

try {

ps=conn.prepareStatement(sql);

} catch (SQLException e1) {

// TODO 自动生成的 catch 块

e1.printStackTrace();

}

try {

ps.setString(1,m.getUsername());

ps.setString(2,m.getPassword());

ps.setString(3, m.getDept());

ps.setString(4, m.getName());

ps.setString(5, m.getScount());

ps.setString(6,m.getUsername());

if(ps.executeUpdate()>0){

flag2=true;

}

if(ps!=null)

ps.close();

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return flag2;

}

//根据用户名删除该用户的信息

public manager doDelete(String username){

manager m=null;

//获取连接

conn=DBUtil.getConn();

//操作数据

String sql="delete from manager where username=?";

try {

ps=conn.prepareStatement(sql);

ps.setString(1,username);

if(ps.executeUpdate()>0){

return m;

}

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return m;

}

}

④修改Servlet

//将用户名密码封装到LoginBean中

manager m=**new** manager();

m.setUsername(request.getParameter("username"));

m.setPassword(request.getParameter("password"));

m.setDept(request.getParameter("dept"));

m.setName(request.getParameter("name"));

m.setScount(request.getParameter("scount"));

ManagerDao mDao=**new** ManagerDao();

**if**(mDao.doUpdate(m)){

//注册信息,将信息放到session中

request.getSession().setAttribute("username",request.getParameter("username"));

request.getSession().setAttribute("password",request.getParameter("password"));

request.getSession().setAttribute("dept",request.getParameter("dept"));

request.getSession().setAttribute("name",request.getParameter("name"));

request.getSession().setAttribute("scount",request.getParameter("scount"));

//服务器跳转到userLoginPage.jsp

//注册成功

request.getRequestDispatcher("manShowPage.jsp").forward(request,response);

}

**else**{

//注册错误信息

request.setAttribute("error", "修改错误，请重新输入");

//服务器跳转到login.jsp

request.getRequestDispatcher("updateInfoPage.jsp").forward(request,response);

}