基于 J2EE 的图书馆管理系统

摘要

随着信息社会的高科技化、商品经济的高效率化和网络的飞速发展，计算机的应用已经普及到经济和社会生活的各个领域，互联网已日益成为管理收集提供信息的最佳渠道并进入传统的流通领域，成为当今人们生活的不可或缺的一部分，人们通过计算机网络对图书馆的管理已成为一种必然。基于这些，本文开发了这套图书管理系统，主要意义有：

1．为了对图书馆的图书管理更加方便安全。

2．为了图书馆管理员管理更方便。

3．为了减低管理运营费用和维护成本。

4．通过本系统软件，能帮助管理员利用浏览器快速方便的进行对图书馆的管理。

本系统采用现在流行的 J2EE 编程、Eclipse3.2 开发和采用 sqlserver 数据库。

关键词：图书馆管理； JAVA； sqlserver 数据库； JavaScript； Eclipse

Library management system based on J2EE

Abstract

Because of the information society's high technology, the commodity economy high efficiency, the network rapid development, the computer application has popularized to the economical and social life domain, the network already became an indispensable part in people's life. In recent years, with the rise of the internet, it has become the best channel to provide the information for the management collection, and enters traditional distribution realm. The computer network and people's daily life relations is increasingly close, the people have become one fashion tidal current through the computer to the library management.

The main significance developing this set of systematic is :

1. In order to manage the library more convenient and secure.
2. In order to make librarian easy.
3. In order to decrease the management operation expense and the maintenance cost.
4. This system software can help the manager using browser fast convenience carrying on to the library management.

This system uses the present popular J2EE to programm, Eclipse3.2 to develop, and uses the SQLSERVER database.

Key words: Library management; JAVA; Sqlserver database;JavaScript;Eclipse

目录

1.系统开发背景

1.1 系统开发背景

在我国现代社会中,随着市场竞争的日益激烈和客户价值选择的变迁,企业越来越意识到争取市场、赢得并保留客户的重要性,这使得处于竞争大潮中的企业不得不开始重视图书馆管理的研究。为客户提供优异的客户价值是企业竞争优势的根本所在,因此读者关系管理在图书馆管理中的地位也变得越来越重要。

现阶段的优缺点是图书馆作为一种信息资源的集散地，图书和用户借阅资料繁多，包含很多的信息数据的管理，现今，在中国有很多的图书馆都是初步开始使用，甚至尚未使用计算机进行信息管理。根据调查得知，他们以前对信息管理的主要方式是基于文本、表格等纸介质的手工处理，对于图书借阅情况（如借书天数、超过限定借书时间的天数）的统计和核实等往往采用对借书卡的人工检查进行，对借阅者的借阅权限、以及借阅天数等用人工计算、手抄进行。数据信息处理工作量大，容易出错；由于数据繁多，容易丢失，且不易查找。总的来说，缺乏系统，规范的信息管理手段。尽管有的图书馆有计算机，但是尚未用于信息管理，没有发挥它的效力，资源闲置比较突出，这就是管理信息系统的开发的基本环境。

在中国现阶段，数据处理手工操作，工作量大，出错率高，出错后不易更改。图书馆采取手工方式对图书借阅情况进行人工管理，由于信息比较多，图书借阅信息的管理工作混乱而又复杂；一般借阅情况是记录在借书证上，图书的数目和内容记录在文件中，图书馆的工作人员和管理员也只是当时对它比较清楚，时间一长，如再要进行查询，就得在众多的资料中翻阅、查找了，造成查询费时、费力。如要对很长时间以前的图书进行更改就更加困难了。基于这此问题，我认为有必要将图书管管理系统朝着图书管理工作规范化，系统化，程序化发展，避免图书管理的随意性，提高信息处理的速度和准确性，能够及时、准确、有效的查询和修改图书情况。

现状是我国大多数的研究人员由于对 JAVA 的应用经验还不足,实施效果不尽如人意。许多企业的高层领导认为图书馆管理系统的实施就是把硬件软件购买回来，认为,这样就解决了管理的所有问题。另外,还有领导认为图书馆管理系统就是客户数据库的应用，没有将图书馆管理系统的功能完全发挥出来。

未来的发展趋势主要是基于 Web 的体系结构：将数据存取建立在 J2EE 或者

NET 标准的应用服务器之上，并且充分利用其中内置的先进技术；个性化的信息入口：新开发的图书馆管理系统软件应该能为用户提供操作便捷的集成化信息人口，帮助用户以自己习惯的界面访问来自不同应用系统的信息；另外，它应当能够支持用户对这个界面进行自我设计，并提供数据流向控制和数据分析方法的定制，从而成为访问和利用 Web Service 的重要途径；分析型的业务过程：利用数据挖掘工具帮助读者从海量的图书资料中寻找潜在的、有价值的信息，从而确保读者活动体贴周到、令人满意。

1.2 基于 WEB 的概念

基于 WEB 就是应用目前比较广泛使用的 B/S 模式（browser/server）,B/S 结构是真正的三层结构，其结构组成如图 1-1 所示：

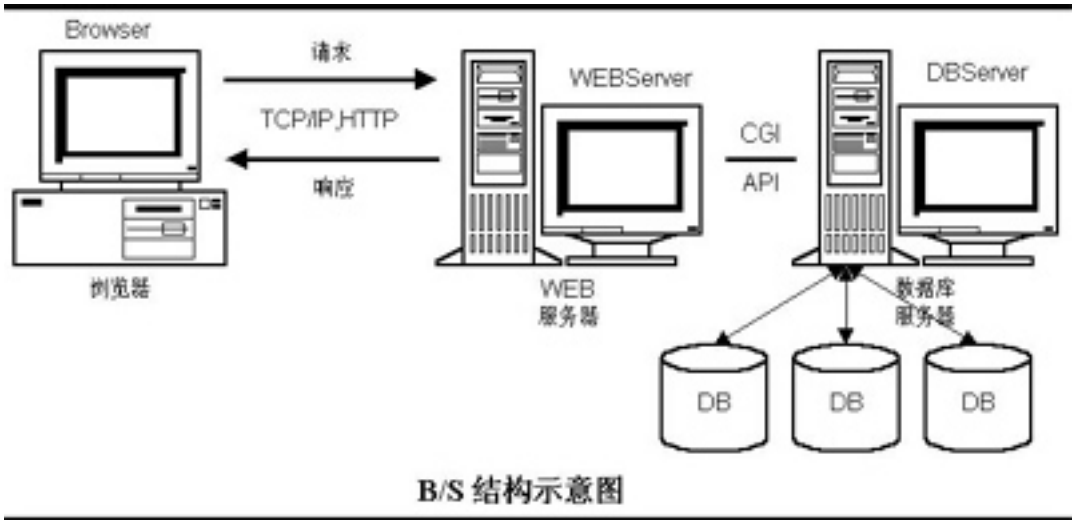


图 1-1 B/S 结构

第一层浏览器是表示层，完成用户接口功能，在客户端向指定的服务器发出请求，WEB 服务器用 HTTP 协议把所需的文件资料传给用户，客户端接受并显示在 WWW 服务器上。

第二层 WEB 服务器是功能层，完成客户的应用功能，即 WEB 服务器接受客户的请求，以 j2ee 与数据库连接，进行申请处理，而后数据库结果返回服务器，再传至客户端。

第三层数据库服务器是数据层，数据库服务器应客户请求进行各种数据处理。

B/S 模式的优点和缺点

1. B/S 模式的优点

2. 具有分布性特点，可以随时随地进行查询、浏览等业务处理。

3. 业务扩展简单方便，通过增加网页即可增加服务器功能。

4. 维护简单方便，只需要改变网页，即可实现所有用户的同步更新。

5. 开发简单，共享性强。

B/S 模式的缺点

1. 个性化特点明显降低，无法实现具有个性化的功能要求。

2. 操作是以鼠标为最基本的操作方式，无法满足快速操作的要求。

3. 页面动态刷新，响应速度明显降低。

1.3 Eclipse 的概述

Eclipse 是一个功能完整且成熟的开发环境，是由蓝色巨人 IBM 所释出。IBM 花了 4 千万美金来开发这个 IDE(Integrated Development Environment)。第一版 1.0 在 2001 年 11 月释出，随后逐渐受到欢迎。

Eclipse 是一个开放源代码的、与 NetBeans、Sun ONE Studio 和 Borland Jbuilder 类似的一种基于 Java 的整合型可扩展开发平台，也是目前最著名的开源项目之一，IBM 在最近几年里也一直在大力支持该项目的发展，目标是将其做成用以替代 IBM Visual Age for Java（简称 IVJ）的下一代 IDE 开发环境，并于 2001 年 11 月宣布投入 4 千万美元资金到该项目的研发。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。幸运的是，Eclipse 附带了一个标准的插件集，包括 Java 开发工具（Java Development Tools，JDT）。其未来的目标不仅仅是成为专门开发 Java 程序的 IDE 环境，根据 Eclipse 的体系结构，通过开发插件，它能扩展到任何语言的开发，甚至能成为图片绘制的工具。

目前，Eclipse 已经开始提供 C 语言开发的功能插件。更难能可贵的是，Eclipse 是一个开放源代码的项目，任何人都可以下载 Eclipse 的源代码，并且在此基础上开发自己的功能插件。也就是说未来只要有人需要，就会有建立在 Eclipse 之上的 COBOL，Perl，Python 等语言的开发插件出现。同时可以通过开发新的插件扩展现有插件的功能，比如本系列文章为了进行手机应用程序的开发就是通过 J2ME 插件的扩展来加以实现的。可以无限扩展，而且有着统一的外观，操作和系统资源管理，这也正是 Eclipse 的潜力所在。

2.系统设计

2.1 设计目标

根据图书馆日常图书管理工作的需求和图书借阅的管理流程，该系统实施后，应该达到以下目标：

1．界面设计友好，美观，数据存储安全，可靠。

2．基本信息设置保证的图书信息和读者信息的分类管理。

3．实现了图书信息管理和读者信息管理。

4．强大的查询功能，保证了数据查询的灵活性。

5．实现对图书借阅，续借，归还过程的全程数据信息跟踪。

6．提供借阅到期提醒功能，使管理者可以及时了解到已经到达归还日期的图书借阅信息。

7．提供管理员修改自己密码的功能，保证系统的安全性。

8．系统最大限度地实现了易维护性和易操作性。

9．提供灵活，方便的权限设置功能，使整个系统的管理分工明确。

10．采用人机对话的操作方式，方便管理员的日常操作。

11．基于 j2ee 技术的图书管理系统的分析与设计是立足于当今的网络系统的发展趋势，本系统采用了目前企业开发应用中最流行的组合，系统可扩展性强，性能高。

12.本系统采用目前最流行的 J2EE+Sqlserver 进行系统的开发，采用了 MD5 加密算法，利用 Sqlserver 对 Java 的强大支持，以及 Java 的各种优点，能够在安全性、扩展性、效率性等各方面得到提升。

2.2 系统可行性分析

根据调查得知，以前的图书馆采取手工方式对图书借阅信息进行管理。将一般的借阅情况记录在借阅证上，将图书的数目和内容记录在文件中，这样图书馆的工作人员只能是对当前的借阅信息比较清楚，时间一长，再进行查询时，就得在众多的资料中翻阅，查询了，即费时又费力。比如对很长时间以前的图书信息进行更改就更加困难了。手工操作使得图书借阅信息的管理工作混乱而又复杂。

基于这些问题，有必要建立一个图书管理系统，使图书馆的日常管理工作规

范化，系统化，程序化，避免管理的随意性，提高信息处理的速度和准确性，能够及时，准确，有效地查询图书借还情况。

2.3 开发及运行环境

硬件平台：

CPU：P41.8GHZ

内存：512MB 以上

软件平台：

操作系统：Windows xp‘

数据库：SQLSERVER5

开发工具包：JDK Version 1.5.0

JPS 服务器：Tomcat 5.5

浏览器：IE6.0 及以上版本

分辨率：最佳效果 1024 像素*768 像素

3.数据库的分析与设计

数据库设计的目标是要求完全满足业务的数据存储要求。如果能够设计一个合理的数据库模型，不仅会降低程序编程和维护的难度，也会提高系统实际运行的性能，因而必须仔细的制定设计步骤方案，了解规范的设计方法和必要的规则。

3.1 数据库设计原则

在数据库的设计中，首先要注意命名的规范，其次就是要注意数据的一致性和完整性，尽可能的降低数据的冗余，当然如果数据冗余度低，数据的完整性容易得到保证，但增加了表间连接查询的操作，所以合理的数据冗余也是必要的。可使用规则和约束来对数据的有效性验证。另外可以创建索引，来维护被索引列的唯一性和提供快速访问表中数据的策略。

选择合适的数据库是项目开发成功与否的先决条件，我们在设计时应该从以下几方面去考虑数据库的选择：

1.易用性

2.分布性

3.并发性

4.数据完整性

5.安全性

6.数据恢复性

3.2 概念结构设计

将需求分析得到的需求抽象为信息结构即概念模型的过程就是概念结构设计。概念结构独立于数据库逻辑结构，也独立于支持数据库的 DBMS。它是现实世界与机器世界的中介，它一方面能够充分反映现实世界，包括实体与实体之间的联系，同时又易于向关系、网状、层次等数据模型转换，它是现实世界的一个真实模型，当现实世界需求改变时，概念结构也可以很容易地作相应调整，因此概念结构设计是整个数据库设计的关键所在。

概念结构通常有自顶向下、自底向上、逐步扩张、混合策略四类方法。通常采用的策略是自底向上方法，即自顶向下进行需求分析，然后再自底向上地设计

概念结构。但无论采用哪种设计方法，一般都以 E-R 模型为工具来描述概念结构。

但本系统结构若用 ER 图来描述，将比较繁琐，不够直观。现在就用系统结构来描述，如图 3-1 所示：

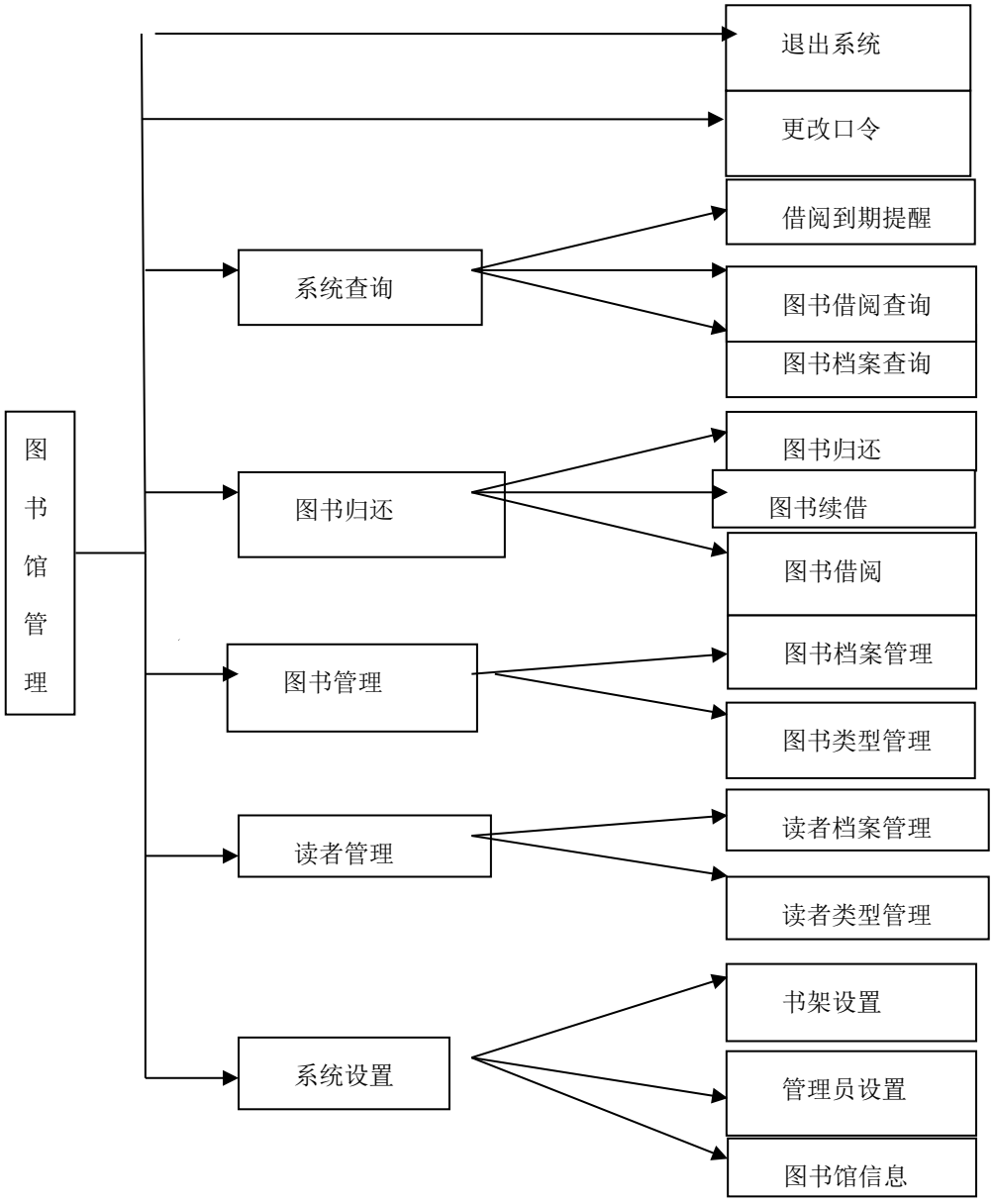


图 3-1 系统结构图

3.3 逻辑结构设计

设计逻辑结构的是和分三步进行：

1. 将概念结构（上图）转化为一般关系模型

2. 将转化来的关系模型向 Sqlserver 支持下的数据模型转化

3. 对数据模型进行优化，消除冗余字段。对数据依赖进行极小化处理。

对关系模式进行必要的分解合并和优化。

对于数据库的优化处理具体表现有：表与表之间设立了主外键关系，尽量减少表的数量，更简介明了的阐述了各表之间的联系。同时也尽可能的减少各表中的多余字段。其中，借阅图书表中引用了用户表的 ID 为外键和图书表中的图书 ID 为外键，这样有效的把 3 张表更紧密的联系起来，同时也对数据模型进行了优化。

本系统数据库采用 SQLSERVER 数据库，系统数据库名称为 db_library。下面给出数据表概要说明及数据表的结构。

主要数据表的结构

tb_library(图书馆信息表)：图书馆信息表主要用来保存图书馆的基本信息。其结构如表 3-1 所示：

表 3-1 图书馆信息表

字段名	数据类型	是否为空	是否主键	默认值	描述
ID	int		主键	NULL	ID
Libraryname	Varchar(100)	Yes		NULL	馆名
Curator	Varchar(100)	Yes		NULL	馆长
Tel	Varchar(100)	Yes		NULL	联系电话
Address	Varchar(100)	Yes		NULL	联系地址
Email	Varchar(100)	Yes		NULL	联系邮箱
url	Varchar(100)	Yes		NULL	网址
Creatdate	Date	Yes		NULL	建馆日期
introduce	text	Yes		NULL	简介

Tb_manager(管理员信息表)：管理员信息表主要用来保存管理员信息。其结构如表 3-2 所示：

表 3-2 管理员信息表

字段名	数据类型	是否为空	是否主键	默认值	描述
ID	Int		pri	NULL	自动编号
Name	Varchar(100)	Yes		NULL	管理员名称
Pwd	Varchar(100)	Yes		NULL	密码

Tb_purview(权限表): 权限表主要用来管理员的权限信息,该表中的 ID 字段与管理员信息表中的 id 字段相关联.其结构如表 3-3 所示:

表 3-3 权限表

字段名	数据类型	是否为空	是否主键	默认值	描述
ID	int		PRI	0	管理员 ID 号
sysset	int	Yes		0	系统设置
Readset	int	Yes		0	读者管理
Bookset	int	Yes		0	图书管理
Borrowback	int	Yes		0	图书借还
sysquery	int	Yes		0	系统查询

Tb_booktype(图书类型表):图书类型表主要用来保存图书类型信息.其结构如表 3-4 所示:

表 3-4 图书类型表

字段名	数据类型	是否为空	是否主键	默认值	描述
ID	Int		pri	null	自动编号
typename	varchar	Yes		null	类型名称
days	Int	Yes		null	可借天数

Tb_bookcase(书架信息表):书架信息表主要用来保存图书信息.其结构如表 3-5 所示:

表 3-5 书架信息表

字段名	数据类型	是否为空	是否主键	默认值	描述
ID	Int		pri	null	自动编号
Name	varchar	Yes		null	书架名称

Tb_bookinfo(图书信息表):图书信息表主要用来保存图书信息.其结构如表 3-6 所示:

表 3-6 图书信息表

字段名	数据类型	是否为空	是否主键	默认值	描述
Barcode	varchar	Yes	Pri	null	条型码
Bookname	varchar	Yes		null	书名
Typied	Int	Yes		null	类型
Author	varchar	Yes		null	作者
Translator	varchar	Yes		null	译者
ISBN	varchar	Yes		null	出版社
Price	Float	Yes		null	价格
Page	Int	Yes		null	页码
Bookcase	Int	Yes		null	书架
storage	Int	Yes		null	库存量
Intime	Date	Yes		null	录入时间
Operator	varchar	Yes		null	操作员
Del	Int	Yes			是否删除
ID	Int		Pri	null	自动编号

Tb_borrow(图书借阅信息表):图书借阅信息表主要用来保存图书借阅信息.其的结构如 3-7 所示:

表 3-7 图书借阅信息表

字段名	数据类型	是否为空	是否为空	默认值	描述
ID	Int		Pri	null	自动编号
Readerid	Int	Yes		null	读者编号
Bookid	Int	Yes		null	图书编号
Borrowtime	Date	Yes		null	借书时间
Backtime	Date	Yes		null	应还时间
Operator	Varchar	Yes		null	操作员
Ifback	Int	Yes		null	是否归还

Tb_giveback(图书归还信息表): 图书归还信息表主要用来保存图书归还信息.其结构如 3-8 所示:

表 3-8 图书归还信息表

字段名	数据类型	是否为空	是否为空	默认值	描述
ID	Int		pri	null	自动编号
Readerid	Int	Yes		null	读者编号
Bookid	Int	Yes		null	图书编号
Backtime	Date	Yes		null	归还时间
Operator	Varchar	Yes		null	操作员

Tb_publishing(出版社信息表): 出版社信息表主要用来保存出版社信息.其结构如 3-9 所示:

表 3-9 出版社信息表

字段名	数据类型	是否为空	是否主键	默认值	描述
ISBN	varchar	Yes		null	ISBN 号
pubname	varchar	Yes		null	出版社名称

Tb_readr(读者信息表): 读者信息表主要用来保存读者信息.其结构如 3-10 所示:

表 3-10 读者信息表

字段名	数据类型	是否为空	是否主键	默认值	描述
ID	Int		Pri	null	自动编号
Name	Varchar	Yes		null	姓名
Sex	Varchar	Yes		null	性别
Barcode	Varchar	Yes		null	条型码
vocation	Varchar	Yes		null	职员
Birthday	Date	Yes		null	出生日期
Papertype	Varchar	Yes		null	有效证件
Paperno	Varchar	Yes		null	证件号码
Tel	Varchar	Yes		null	电话
Email	Varchar	Yes		null	电子邮件
Creatdate	Date	Yes		null	登记日期
Operator	Varchar	Yes		null	操作员
Remark	Text	Yes		null	备注
typeid	Int	Yes		null	类型

Tb_readtype(读者类型信息表): 读者类型信息表主要用来保存读者类型信息. 其结构如 3-11:

表 3-11 读者类型信息表

字段名	数据类型	是否为空	是否主键	默认值	描述
ID	Int		Pri	null	自动编号
Name	Varchar	Yes		null	名称
Number	Int	Yes		null	可借数量

3.4 数据库创建

数据库的创建，主要是借助于 Sqlserver 的 GUI 工具生成，它可以以视图的形式来创建数据库和表，并自动生成相应的 SQL 语句。

（1）进入到 Sqlserver 的程序目录/sqlserver/bin 中，运行 winsqlserveradmin.exe 程序，启动 SQLSERVER 服务器管理界面。

（2）在运行窗口中输入“cmd”命令进入到 DOS 窗口，键入“sqlserver”命

令进入 Sqlserver 语言输入界面，键入“exit”命令退出 sqlserver。

（3）键入“creat database db_library;”命令即可创建 db_library 数据库。

4. 系统基本功能实现

4.1 管理员功能模块

管理员功能模块包括以下功能：

1．管理员登陆：用于登陆系统。

2．添加管理员信息：用于添加管理员。

3．查看管理员列表：用于查询并显示系统中的全部管理员及其权限信息。

4．管理员权限设置：用于设置或修改管理员权限。

5．管理员删除：用于删除系统中的管理员信息及其权限信息。

6．更改口令：用于管理员登陆后修改自己的密码。

4.1.1 管理员实现类设计

在管理员模块中，涉及的数据表是管理员信息表（tb_manager）和权限表(tb_purview),在管理员信息表中保存着管理员名称及密码等信息，在权限表中保存着各管理员的权限信息，这两个表通过各自的 id 字段相关联。通过这两个表可以获得完整的管理员信息，根据这些信息来创建管理员模块的 ACTIONFORM，名称为“Managerform”.

STRUTS 的核心在于他的 ACTION，一般在 ACTION 里面做对页面的跳转工作。管理员功能模块的 ACTION 实现类继承了 ACTION 类，在该类中首先需要在该类的构造方法中实例化管理员模块的 MANAGERDAO 类。ACTION 实现类

的主要方法是 PERFORM（），该方法会被自动执行，这个方法本身没有具体的事务，他是通过 HTTPSERVLETREQUEST 的 GETPARAMETER（）方法获取的 ACTION 参数执行相应的方法的。

4.1.2 系统登陆设计

系统的登陆模块是图书馆管理系统中最先使用的功能，是系统的入口。在系统登陆页面中，系统管理人员可以通过输入正确的管理员名称和你密码进入到系统，当用户没有输入管理员名称或密码时，系统会通过 JAVASCRIPT 进行判断，并给予信息。登陆界面如图 4-1 所示：

图 4-1 登陆界面

系统登陆界面主要用于收集管理员的输入信息及通过自定义的 Javascript 函数验证输入信息是否为空，该页面中涉及的表单元素如表所示。

名称	元素类型	重要属性	含义
Form1	Form	Method=”post” action=”manader.do?action=login”	管理员登陆表单
Name	Text	Size=”25”	管理员名称
Pwd	Password	Size=”25”	管理员密码
Submit	Submit	Value=”确定”onclick=”return check(form1)”	“确认”按钮
Submit3	Reset	Value=”重置”	“重置”按钮
Submit2	button	Value=”关闭” onclick=” window.close();”	“关闭”按钮

其中“确定”按钮的命令代码，验证用户身份是否合法的设计思路为：首先判断用户名和密码是否为空，如果为空，则提示用户输入用户名和密码，否则以用户名和密码为条件，从数据库中查询数据，有数据返回，证明用户名身份合法；反之，身份不合法。

4.1.3 管理员设置

一．查看管理员列表

管理员登陆后，选择“系统设置”/“管理员设置”菜单项，进入到查看管理员列表页面，在该页中，将列出系统中全部管理员及其权限信息，同时提供添加管理员信息，删除管理员信息，设置管理员权限的超级链接。查看管理员列表页面的运行结果如图 4-2 所示。

图 4-2 查看管理员列表页面的运行结果

设计思路：查看管理员列表模块涉及的 ACTION 的参数为”managerQuery”,当 action= managerQuery 时,就会调用查看管理员列表的方法 managerQuery().

在方法 managerQuery()中,首先调用 managerdao 类中的 Query()方法查询全部管理员信息,在将返回的查询结果保存到 HttpServltRequest 的对象 managerQuery 中。

二．添加管理信息

管理员登陆后，选择“系统设置”/“管理员设置”菜单项，进入到查看管理员列表页面，在该页面中单击“添加管理信息”超级链接,打开添加管理员信息页面.添加管理员信息页面的运行结果如图 4-3 所示.

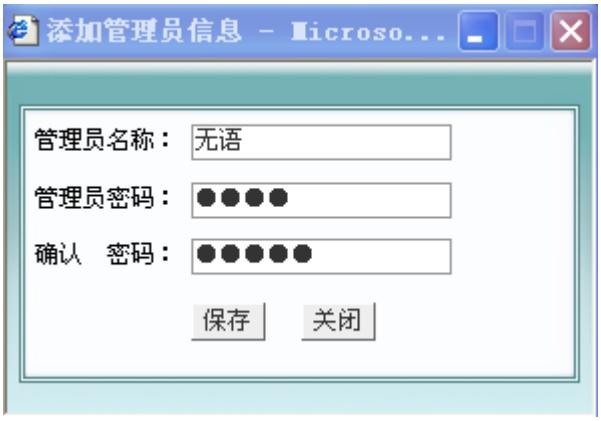


图 4-3 添加管理员信息

设计思路:在添加管理员信息的方法 managerAdd()中,首先需要将收到的表单信息强制转换成 ActionForm 类型,并用获得指定属性的 getXXX 方法重新设置该属性的 setXXX 方法,然后调用 managerDAO 类中的 insert()方法,将添加的管理员信息保存到数据表中,并将返回值保存到变量 ret 中,如果返回值为 1,表示信息添加成功,将页面重新定向到添加信息成功页面;如果返回值为 2,表示该管理员信息已添加,将错误提示信息”该管理员信息已存在!”保存到 HttpServltRequest 的对象 error 中,然后将页面重新定向到错误提示信息页面;否则,将错误提示信息”添加管理员信息失败!”保存到 HttpServltRequest 的对象 error 中,并将页面重定向到错误提示页.

三. 权限设置

管理员登陆后，选择“系统设置”/“管理员设置”菜单项，进入到查看管理员列表页面，在该页面中单击”权限设置”超级链接,进入到”设置管理员权限”页面.设置管理员权限页面的运行结果如图 4-4 所示.

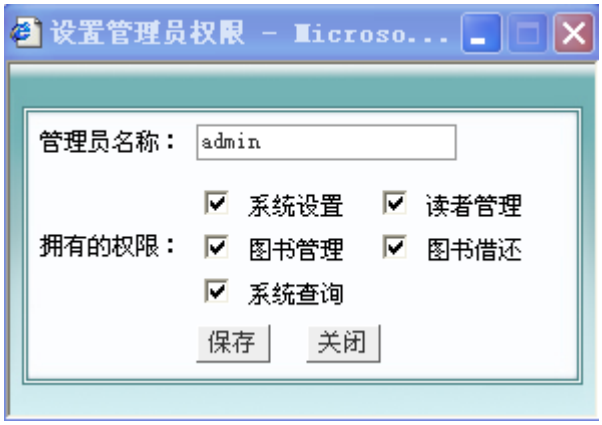


图 4-4 设置管理员权限页面的运行结果

设计思路:在查询指定管理员权限信息的方法 managerModifyQuery()中,首先需要将接收到的表单信息强制转换成 ActionForm 类型,并用获得指定属性 getXXX

方法重新设置该属性的 setXXX 方法,再调用 ManagerDAO 类中的 query_update()方法,查询出指定的管理员权限信息,再将返回的查询结果保存到 HttpServltRequest 的对象 managerQueryif 中.

四. 删除设计

管理员登陆后，选择“系统设置”/“管理员设置”菜单项，进入到查看管理员列表页面，在该页面中单击想要删除的管理员信息后面的“删除” 超级链接，该管理员信息即被删除.

设计思路:在删除管理员的方法 managerDel()中,首先需要将接收到的表单信息强制转换成 ActionForm 类型,并用获得的 id 参数的值重新设置该 ActionForm 的 setID()方法,再调用 ManagerDAO 类中的 delete()方法,删除指定的管理员,并根据执行结果将页面转到相应页面.

4.2 图书档案管理功能模块设计

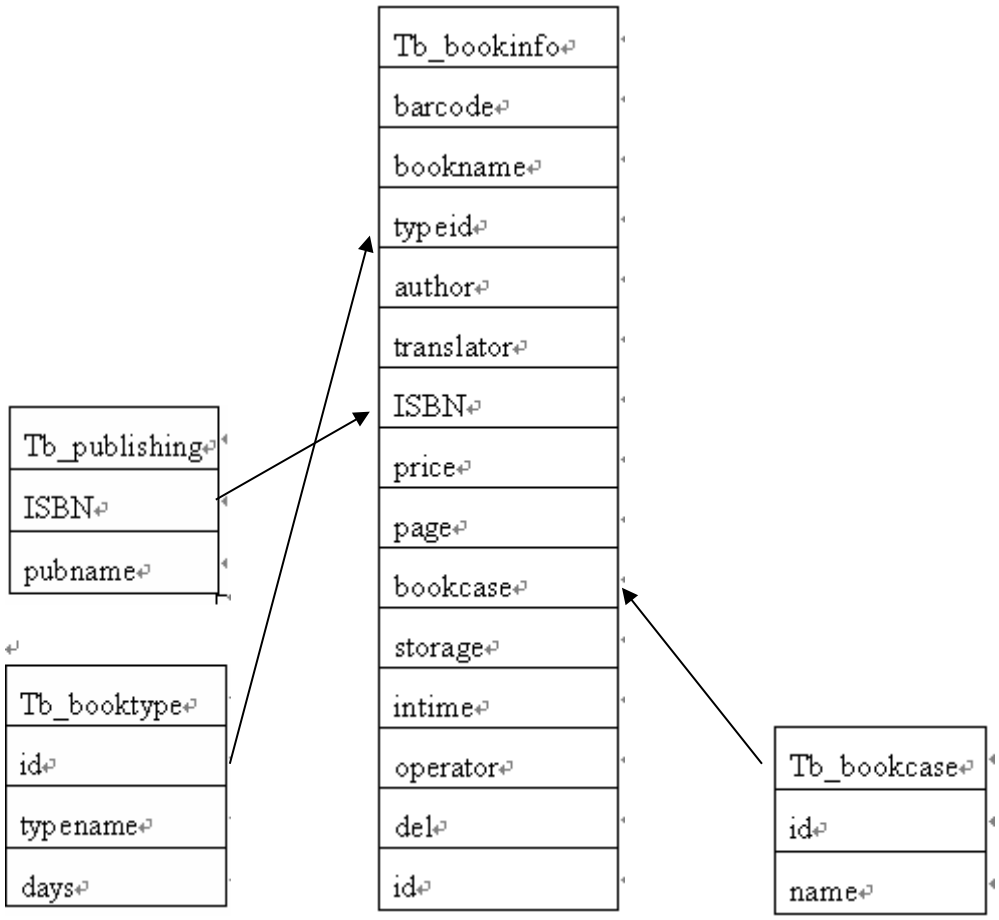
图书档案管理功能模块包括以下功能：

1. 查看信息列表：用于显示图书的基本信息。
2. 图书详细信息：用于指定图书的详细信息。
3. 添加图书信息。
4. 删除图书信息。
5. 查询图书信息：用于不同条件查询不同信息。

4.2.1 图书管理实现类设计

在图书档案管理模块中涉及的数据表是图书信息表（tb_bookinfo），书架设置表（tb_bookcase）,图书类型表（tb_booktype）和出版社信息表（tb_publishing），这四个数据表间通过相应的字段进行关联.其表如表 4-1 所示.

表 4-1 图书档案管理模块各表间关系图



通过以上 4 个表可以获得完整的图书档案信息，根据这些信息来创建图书档案模块的 ActionForm，名称为“BookForm”。

图书档案功能模块的 ACTION 实现类 BOOK 继承了 ACTION 类。在该类中，首先需要在该类的构造方法中实例化图书档案模块的 BOOKDAO 类。ACTION 实现类的主要方法是 PERFORM（），该方法会被自动执行，他本身没有具体的事务，他是根据通过 HTTPSERVLETREQUEST 的 GEIPARAMETER（）方法获取的 ACTION 参数值执行相应方法的。

4.2.2 图书管理设计

一．查看图书列表

管理员登陆后，选择“图书管理”/“图书档案管理”菜单项，进入到查看图书列表页面，在该页面中将显示全部图书信息列表，同时提供添加图书信息，删除图书信息，修改图书信息的超链接。查看图书信息列表页面的运行结果如图 4-5 所示。

图 4-5 查看图书信息列表页面

设计思路：在查看图书信息列表的方法 bookQuery()中，首先调用 BookDAO 类中的 query()方法查询全部图书信息,再将返回的查询结果保存到 HttpServltRequest 的对象 book 中.

二．添加图书信息

管理员登陆后，选择“图书管理”/“图书档案管理”菜单项，进入到查看图书列表页面，在该页面中单击添加的图书信息后面的添加，进入到添加图书信息页面。

添加图书信息界面如图 4-6 所示：

图 4-6 添加图书信息界面

在添加图书信息的方法 bookAdd()中,首先需要将接收到的表单信息强制转换成 ActionForm 类型， 并用获得指定属性 getXXX 方法重新设置该属性的 setXXX

方法,然后调用 BookDAO 类中的 insert()方法,将添加的图书信息保存到数据表,并将返回值保存到 ret 中:如果返回值为 1,表示添加信息成功,将页面重定向到添加信息成功页面;如果返回值为 2,表示该图书信息已经添加,将错误信息提示“该图书信息已经添加!”保存到 HttpServltRequest 的对象 error 中,然后将页面重定向到错误信息提示页面;否则将错误提示信息“图书信息添加失败!”保存到 HttpServltRequest 的对象 error 中,并将页面重定向到错误提示页.

三. 修改图书信息

管理员登陆后，选择“图书管理”/“图书档案管理”菜单项，进入到查看图书列表页面，在该页面中单击修改的图书信息后面的修改，进入到修改图书信息页面。

设计思路:此思路与添加图书信息的设计思路相似.区别就是用到的方法为 bookModify(),调用的是 BookDAO 类中的 update()方法,和修改的图书信息保存在数据表 tb_bookinfo 中.

四. 删除图书信息

管理员登陆后，选择“图书管理”/“图书档案管理”菜单项，进入到查看图书列表页面，在该页面中单击想要删除的图书信息后面的删除，进入到删除图书信息页面。

设计思路:在删除图书信息的方法 bookDel()中,首先需要将接收到的表单信息强制转换成 ActionForm 类型,,并用获得的 id 参数的值重新设置该 ActionForm 的 setID 方法,再调用 BookDAO 类中的 delete()方法删除指定的图书信息，并根据执行结果将页面转到相应页面。

4.3 图书借还管理功能模块设计

图书借还管理功能模块包括图书借还，系统查询 2 个子模块。其具体功能如下：

1. 图书借还模块包括图书借阅，图书归还和图书续借 3 个子模块。
2. 系统查询模块包括图书借阅查询和借阅到期提醒 2 个子模块。

4.3.1 图书借还实现类设计

在图书借还管理模块中涉及的数据表是图书借阅信息表（tb_borrow）,图书信息表（tb_bookinfo）和读者信息表（tb_read），这 3 个数据表通过相应的字段进行关联。根据这些表来创建图书借还模块的 ActionForm，名称为“borrowform”.

图书借还管理模块的 ACTION 实现类 BORROW 继承了 ACTION 类。在该类中，首先需要在该类的构造方法中实例化图书借还管理模块的 BOOKDAO 类，BORROWDAO 类和 READERDAO 类。ACTION 实现类的主要方法是 PERFORM（），该方法会被自动执行，他本身没有具体的事务，他是根据通过 HTTPSERVLETREQUEST 的 GEIPARAMETER（）方法获取的 ACTION 参数值执行相应方法的。

4.3.2 图书借还设计

一．图书借阅界面

管理员登陆后，选择“图书借还”/“图书借阅”菜单项，进入到图书借阅页面，在该页面中的“读者条型码”文本框中输入读者的条形码后，单击“确定”按钮，系统会自动检索出读者的基本信息和未归还的借阅图书信息，并将其显示在页面中，此时输入图书的条形码或图书的名称后，单击其后面的“确定”按钮，借阅指定的图书。

图书借阅界面如图 4-7 所示：

图 4-7 图书借阅界面

设计思路：实现图书借阅的方法 bookborrow()需要分以下 3 个步骤进行：

1．首先需要实例化一个读者信息所对应的 ActionForm（readform）的对象，然后将该对象的 setBarcode()方法设置为从页面中获得的条形码值，再调用 ReadDAO 类中的 queryM()方法查询读者信息,并将查询结果保存在 ReadForm 的对象 Reader 中,最后将 Reader 保存到 HttpServltRequest 的对象 readinfo 中.

2.调用 borrowDAO 类的 borrowinfo()方法查询读者的借阅信息,并将其保存到 HttpServltRequest 的对象 borrowinfo 中.

3.首先获取查询条件和查询关键子,如果查询关键字不为空,调用 BookDAO 类中的 queryB()方法查询图书信息,当存在符合条件的图书信息时,再调用 BoorrowDAO 类中的 insertBorrow()方法添加图书借阅信息,否则将错误信息提示

“没有该图书”保存到 HttpServltRequest 的对象 error 中.

二. 图书归还

管理员登陆后，选择“图书借还”/“图书归还”菜单项，进入到图书归还页面，在该页面中的“读者条型码”文本框中输入读者的条形码后，单击“确定”按钮，系统会自动检索出读者的基本信息和未归还的借阅图书信息，并将其显示在页面中，此时单击其要归还图书后面的“归还”按钮，即可将该图书归还。

图书归还界面如图 4-8 所示：

图 4-8 图书归还界面

设计思路：实现图书归还的方法 bookback()需要以下 3 个步骤：

1.首先需要实例化一个读者信息所对应的 ActionForm（readform）的对象然后将该对象的 setBarcode()方法设置为从页面中获取的读者条形码的值，再调用 ReadDAO 类中的 queryM()方法查询读者信息，再将查询结果保存在 readform 的对象 reader 中，最后将 reader 保存在 HttpServltRequest 的对象 readinfo 中.

2.调用 BorrowDAO 类中的 borrowinfo()方法，查询读者的借阅信息，并将其保存到 HttpServltRequest 的对象 borrowinfo 中.

3.首先判断是否从页面中传递了借阅的 ID 号，如果是，则获取从页面传递的借阅 ID 号，然后判断该 ID 值是否大于 0，如果大于 0，则调用 BorrowDAO 类中的 back()方法执行读书归还操作.如果图书归还操作执行成功,则将当前读者条形码保存到 HttpServltRequest 的对象的 error 中,并将页面重定向到错误的提示页.

三. 图书借阅查询

管理员登陆后，选择“系统查询”/“图书借阅查询”菜单项，进入到图书借阅查询页面，在该页面中可以按指定的字段或某一时间段进行查询，同时还可以实现按指定字段及时间段进行综合查询。

图书借阅查询的界面如图 4-9 所示：

图 4-9 图书借阅查询界面

设计思路:在图书借阅查询的方法 borrowQuery()中,首先获取表单元素复选框 flag 的值,并将其保存到字符串数组 flag 中,然后根据 flag 的值组合查询字符串,再调用 BorrowDAO 类中的 borrowQuery()方法,并将返回值保存到 HttpServltRequest 的对象 borrowquery 中。

四. 借阅到期提醒

管理员登陆后，选择“系统查询”/“借阅到期查询”菜单项，进入到借阅到期提醒页面，在该页面中将列出全部已经达到归还日期的图书借阅信息。

设计思路:在借阅到期提醒的方法 bremind()中,首先调用 borrowDAO 类中的 bremind()方法,查询全部已经到达归还日期的借阅信息,再将返回的查询结果保存到 HttpServltRequest 的对象 Bremind 中。

4.3.3 关键代码

1.关键字搜索核心代码：

```
String search = this.model.getSearch();
String pageSize = ServletActionContext.getRequest().getParameter("pageSize");
String pageNo = ServletActionContext.getRequest().getParameter("pageNo");
int ps = 4;
int pn = 1;
if ((pageSize != null) && (pageSize.length() > 0)) {
  ps = Integer.parseInt(pageSize);
}
if ((pageNo != null) && (pageNo.length() > 0)) {
  pn = Integer.parseInt(pageNo);
```

```
        }
        List bookList = this.bookService.getClassifyBook((pn - 1) * ps, ps,
"from Books where author like '%" + search + "%' or " +
          "name like '%" + search + "%' or " + "introduction like '%" + search
+ "%'");
        int totalcount = this.bookService.getCount("select count(*) from Books
where author like '%" + search + "%' or " +
          "name like '%" + search + "%' or " + "introduction like '%" + search
+ "%'");
        int count = (totalcount - 1) / ps + 1;
        ServletActionContext.getRequest().setAttribute("search", search);
        ServletActionContext.getRequest().setAttribute("bookList", bookList);
        ServletActionContext.getRequest().setAttribute("pageSize",
Integer.valueOf(ps));
        ServletActionContext.getRequest().setAttribute("pageNo",
Integer.valueOf(pn));
        ServletActionContext.getRequest().setAttribute("count",
Integer.valueOf(count));
```

2.Md5 加密核心代码：

```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;


public class Mdt5 {
    public static void main(String[] arg) {
        String str = "admin";
        MessageDigest messageDigest = null;
        try {
```

```
            messageDigest = MessageDigest.getInstance("MD5");
            messageDigest.reset();
            messageDigest.update(str.getBytes("UTF-8"));
        } catch (NoSuchAlgorithmException e) {
            System.out.println("NoSuchAlgorithmException caught!");
            System.exit(-1);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        byte[] byteArray = messageDigest.digest();
        StringBuffer md5StrBuff = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            if (Integer.toHexString(0xFF & byteArray[i]).length() == 1)
                md5StrBuff.append("0").append(Integer.toHexString(0xFF &
byteArray[i]));
            else
                md5StrBuff.append(Integer.toHexString(0xFF &
byteArray[i]));
        }
        System.out.println( md5StrBuff.toString());
    }
}
```

3.管理员登录信息代码：

```
package com.user.action;

import com.bookInfo.model.service.BookInfoService;
import com.bookType.model.service.BookTypeService;
import com.common.base.action.BaseAction;
import com.common.constant.Constant;
import com.common.util.Util;
```

```
import com.user.model.dao.User;
import com.user.model.dao.UserEntity;
import com.user.model.service.UserService;

public class UserAction extends BaseAction {

    private static final long serialVersionUID = 5397058623582673843L;

    private UserService userService;

    private BookInfoService bookInfoService;

    private BookTypeService bookTypeService;

    private User user;

    public UserService getUserService() {
        return userService;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public User getUser() {
        return user;
    }

    public void setUserService(UserService userService) {
        this.userService = userService;
    }
```

```
    public void setBookInfoService(BookInfoService bookInfoService) {
        this.bookInfoService = bookInfoService;
    }

    public void setBookTypeService(BookTypeService bookTypeService) {
        this.bookTypeService = bookTypeService;
    }

    /**
     * 管理员登录
     * */
    public String userLogin() {
        UserEntity userEntity = this.userService.getUserByUnameAndPwd(user);
        if(userEntity == null) {
            super.getRequest().setAttribute("flg", -1);
            return "error";
        }

        super.getSession().put(Constant.SESSION_ADMIN_KEY, userEntity);
        // 得到管理员总数
        super.getSession().put("adminNum", this.userService.getAdminNum());
        // 得到本站共有图书
        super.getSession().put("bookTotal", this.bookInfoService.getBookTotal());
        // 得到图书种类
        super.getSession().put("bookTypeTotal",
this.bookTypeService.getBookTypeTotal());
        // 图书库存
        super.getSession().put("bookStore",
this.bookInfoService.getBookStoreNum());
        // 今日借阅
```

```
//      super.getRequest().setAttribute("adminNum", "");
//      // 今日归还
//      super.getRequest().setAttribute("adminNum", "");
//      // 逾期未归换
//      super.getRequest().setAttribute("adminNum", "");
        return "success";
    }

    /**
     * 添加管理员
     *
     * */
    public String addAdmin() {
        int flg = this.userService.addAdmin(user);
        if(flg == 0 || flg == -1){
            super.getRequest().setAttribute("flg", flg);
            super.getRequest().setAttribute("uname", user.getUserName());
            return "addError";
        }
        return this.adminList();
    }

    /**
     * 管理员列表
     *
     * */
    public String adminList() {
        super.getRequest().setAttribute("userList",
this.userService.getUserList());
        return "userList";
    }
```

```
    /**
     * 管理员密码修改
     * */
    public String adminPwdUpdate() {
        UserEntity                              userEntity                              =
(UserEntity)super.getSession().get(Constant.SESSION_ADMIN_KEY);

    if(userEntity.getPassWord().equals(Util.getMd5Str(user.getOldPassWord()))) {
            user.setUserName(userEntity.getUserName());



    super.getRequest().setAttribute("flg",this.userService.adminPwdUpdate(user));
        } else
            // 旧密码不正确
            super.getRequest().setAttribute("flg", -2);
        return "pwdupd";
    }

    public String loginOut() {
        super.getSession().put(Constant.SESSION_ADMIN_KEY, null);
        return "helpjsp";
    }

    /**
     * 一级管理员修改其他管理员密码
     * */
    public String userPwdUpdate() {
        return null;
    }
```

```
}
```

4.增加会员信息代码：

```
package com.students.action;

import java.util.List;

import com.common.base.action.BaseAction;
import com.common.constant.Constant;
import com.students.model.dao.Students;
import com.students.model.dao.StudentsEntity;
import com.students.model.service.StudentsService;

public class StudentsAction extends BaseAction {

    private static final long serialVersionUID = 7242208740482534907L;

    private Students students;

    private StudentsService studentsService;

    public Students getStudents() {
        return students;
    }

    public void setStudents(Students students) {
        this.students = students;
    }

    public void setStudentsService(StudentsService studentsService) {
        this.studentsService = studentsService;
    }
```

```
    /**
     * 信息添加
     * */
    public String addStudents() {
        int flg = this.studentsService.addStudents(students);
        if(flg == Constant.DO_DATABASE_SUCCESS) {
            return this.listStudents();
        }
        super.getRequest().setAttribute("flg", flg);
        return "error";
    }

    public String listStudents() {
        super.getRequest().setAttribute("studentsList",
this.studentsService.getStudentsList()) ;
        return "studList";
    }
}
```

5.增加书籍核心代码：

```
        public String addBook() {
        if ((this.model.getId() != null) && (this.model.getId().intValue() > 0))
{
          Books book = this.bookService.getBookById(this.model.getId());
          if ((book != null) && (book.getId().intValue() > 0) &&
            (book.getPicture() != null) && (book.getPicture() != "") &&
(book.getPicture().length() > 0) && (
            (this.model.getPicture() == null) || (this.model.getPicture() == "")
|| (this.model.getPicture().length() <= 0))) {
            this.model.setPicture(book.getPicture());
          }
```

```
        }
        if ((this.pic != null) && (this.pic.length() > 0L)) {
          int i = this.picFileName.lastIndexOf(".");
          Calendar c = Calendar.getInstance();
          String filename = "picture/" + c.get(1) + (c.get(2) + 1) + c.get(5) +
            c.get(11)    +    c.get(12)    +    c.get(13)    +    c.get(14)    +
this.picFileName.substring(i);
          try {
            String path = ServletActionContext.getRequest().getRealPath("/");
            String temName = path.replace("\\", "/") + filename;
            System.out.println(temName);
            FileOutputStream fos = new FileOutputStream(temName);
            FileInputStream fis = new FileInputStream(getPic());
            byte[] buf = new byte[1024];
            int len = 0;
            while ((len = fis.read(buf)) > 0) {
              fos.write(buf, 0, len);
            }
            if (fos != null) {
              fos.close();
            }
            if (fis != null) {
              fis.close();
            }
            this.model.setPicture(filename);
          } catch (FileNotFoundException e) {
            e.printStackTrace();
          } catch (IOException e) {
            e.printStackTrace();
          }
        }
```

```
        if (this.bookService.saveOrUpdateBook(this.model))
          return "browseBook";
        return "input";
      }
```

6.得到图书总数及库存代码：

```
package com.bookInfo.model.dao;

import java.util.List;

import com.common.base.model.BaseDao;
import com.common.util.Util;

public class BookInfoDao extends BaseDao {

    /**
     * 得到图书总数
     * */
    public Integer getBookTotal() {
        return (Integer)super.queryForObj("managerBookInfo.getBookTotal", null);
    }

    /**
     * 得到图书库存
     * */
    public Integer getBookStoreNum() {
        return      (Integer)super.queryForObj("managerBookInfo.getBookStoreNum",
null);
    }

    public List<BookInfoEntity> getbookList() {
        return                                Util.checkList((List<BookInfoEntity>)
```

```
super.queryForList("managerBookInfo.getBookList", null));
    }

    public Integer addBook(BookInfo bookInfo) {
        return (Integer)super.saveResultInt("managerBookInfo.addBook", bookInfo);
    }

    public BookInfoEntity getBookEntityById(Integer bookId) {
        return (BookInfoEntity) super.queryForObj("managerBookInfo.getBookById",
bookId);
    }
}
```

7.查询书对象：

```
package com.bookInfo.action;

import java.util.Date;

import com.bookInfo.model.dao.BookInfo;
import com.bookInfo.model.service.BookInfoService;
import com.common.base.action.BaseAction;
import com.common.constant.Constant;
import com.students.model.dao.Students;
import com.user.model.dao.User;
import com.user.model.dao.UserEntity;

public class BookInfoAction extends BaseAction {

    /**
     *
     */
    private static final long serialVersionUID = 53466211268427974l1L;
```

```
private BookInfoService bookInfoService;

private BookInfo bookInfo;

private Students students;

public Students getStudents() {
    return students;
}

public void setStudents(Students students) {
    this.students = students;
}

public BookInfo getBookInfo() {
    return bookInfo;
}

public void setBookInfo(BookInfo bookInfo) {
    this.bookInfo = bookInfo;
}

public Integer getBookId() {
    return bookId;
}

public BookInfoService getBookInfoService() {
    return bookInfoService;
}
```

```
    public Integer bookId;

    public void setBookId(Integer bookId) {
        this.bookId = bookId;
    }

    public void setBookInfoService(BookInfoService bookInfoService) {
        this.bookInfoService = bookInfoService;
    }



    public String bookList() {
        super.getRequest().setAttribute("bookList",
this.bookInfoService.getBookList());
        return "bookList";
    }

    public String bookAdd() {
        Date date = new Date();
        UserEntity                          userEntity                          =
(UserEntity)super.getSession().get(Constant.SESSION_ADMIN_KEY);
        this.bookInfo.setCreateAdmin(userEntity.getId());
        this.bookInfo.setCreateTime(date);
        this.bookInfo.setUpdateAdmin(userEntity.getId());
        this.bookInfo.setUpdateTime(date);
        int flg = this.bookInfoService.addBook(bookInfo);
        return bookList();
    }

    public String toBookOutPage() {
```

```
        // 查询书对象
        super.getRequest().setAttribute("bookEntity",
this.bookInfoService.getBookEntityById(this.bookId));
        return "outBook";
    }

    public String bookOut() {

        return this.bookList();
    }
}
```

8.图书库存管理代码：

```
ackage com.bookLen.model.dao;

import java.sql.SQLException;
import java.util.List;

import com.common.base.model.BaseDao;

public class BookLenDao extends BaseDao {

    /**
     * 今天借出图书总数
     * */
    public Integer getTodayOutBook() {
        return (Integer)super.queryForObj("managerBookLen.managerBookLen", null);

    }

    /**
     * 今日归还图书总数
```

```
 * */
public Integer getTodayInBook() {
    return (Integer)super.queryForObj("managerBookLen.managerBookLen", null);
}

/**
 * 逾期未归还图书总数
 * */
public Integer getOutTimeBook() {
    return (Integer)super.queryForObj("managerBookLen.getOutTimeNum", null);
}

public Integer saveBookLen(BookLen bookLen) {
    try {
        super.startTransaction();
        // 书的存量减 1
        super.updateObj("managerBookInfo.minBookNum", bookLen.getBookId());
        // 借出书
        super.saveResultInt("managerBookLen.saveBookLen", bookLen);

        super.commitTransaction();
        return 1;
    } catch(Exception ce) {
        return 0;
    } finally {
        try {
            super.endTransaction();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```
        }
    }

    public Integer inBook(BookLen bookLen) {
        try {
            super.startTransaction();
            // 书的存量加 1
            super.updateObj("managerBookInfo.plusBookNum", bookLen.getBookId());
            // 状态改为入库
            super.saveResultInt("managerBookLen.updateIsReturn",
bookLen.getId());

            super.commitTransaction();
            return 1;
        } catch(Exception ce) {
            return 0;
        } finally {
            try {
                super.endTransaction();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

    public List<BookLenEntity> getBookLenList() {
        return
(List<BookLenEntity>)super.queryForList("managerBookLen.getBookLenList", null);
    }
}
```

4.4 疑难问题的分析和解决

在本系统中会遇到这样的问题:在借阅图书时,需要自动计算图书的归还日期,而这日期又不是固定不变的,它需要根据系统日期和数据表中保存的各类图书的最多借阅天数来计算,既图书归还日期=“系统日期”+“最多借阅天数”.

在本系统中是这样解决问题的:首先获取系统时间,然后从数据表中查询出该类图书的最多借阅天数,最多计算归还日期.计算归还日期的方法如下:

首先将系统时间中的天取出,然后将其与获取的最多借阅天数相加,再将相加后的天与系统时间中的年_月_连接成一个新的字符串,最后将该字符串重新转换为日期.

5.毕业设计总结

5.1 论文总结

整个毕业设计的学习和演练，我学习到了很多新知，也克服了不少困难。回顾毕业设计的完成过程，我发现了自身的不足。我很高兴自己能在大学的最后一次对自己的考核中有所收获。如开始做毕业设计的时候，我有些拖拖拉拉，总觉得还有时间，但一段时间后，我发现自己进度缓慢，于是我有意识的对自己进度加强管理，效率提高很多。我想做任何事都是一样的，要有耐性、要持之以恒，这样才能达到自己的预期目标。当然我发现自身的不足不止这一点，我会在今后的学习和生活中继续改进和提高自己。我感谢有毕业设计这么一个机会能让我做一次对四年学习生活的总结！

对于项目本身，也存在着一些不足。首先，在项目的开始，由于自身 JAVA 编程基础，特别是在创建 JAVA 的类方面的编程知识浅薄，导致很大一块技术难题是在编程上，在代码实现上也可能存在一些错误或者没有得到优化。其次，从项目的演示可以看出，算法的运算速度有点慢，这是由于占用的内存过大。

5.2 展望

针对以上的不足，可以做以下改进：（1）增强自身 JAVA 编程能力，以便更好利用开发工具实现算法。（2）优化代码，减少变量数目，及时释放内存以缓解某一时刻的内存过大的问题。（3）改进算法，设定多种参数，类；运行时，输入相关参数，来实现对不同类的定义，使算法更具有通用性。

当今社会是竞争的社会，正如软件、算法等需要改进才能适应发展和应用一样，每个人只有不断的改进，提高自己，才能适应社会的发展，才能在竞争的大潮中立足，实现自身价值！

致谢

首先我要深深地感谢我的指导老师P。本文是在老师的精心指导和严格要求下完成的。在整个毕业设计过程中，老师从各方面都给予了极大的关怀和悉心的教导。他那极其认真负责的作风和勤奋严谨的精神将使我在以后的研究工作中受益匪浅。在此论文完成之际谨向老师致以深深的敬意和衷心的感谢。

大学本科的学习生活即将结束，在此，我要感谢所有曾经教导过我的老师。同时，我还要感谢我的同学们，他们在我的毕业设计的过程中也给了我很大的帮助，特别是在 JAVA 编程方面。

感谢我的同学们，四年来他们从学习、生活等各方面给以无微不至的关怀！在学习上认真激烈的探讨，困难时替我分担，高兴时与我共享，是我克服困难和进取的动力！

谨以此文献给所有关心、帮助和支持过我的人们！

参考文献

[1] Ryan Asleson , Nathaniel T.Schutta. Ajax 基础教程[M].北京：人民邮电出版社，2006. 07：75-120

[2] Patrick Lightbody , Jason Carreira.WebWork in Action[M].北京：电子工业出版社，2006. 02:247-273

[3] 张立科.JAVA 信息管理系统开发[M].北京：人民邮电出版社，2008. 10：137-153

[4] 林信良.Spring 技术手册[M].北京：电子工业出版社，2006. 01：63-86

[5] 夏昕，曹晓钢,唐勇.深入浅出 Hibernate[M].北京：机械工业出版社，2006. 07：225-245

[6] Craig Walls , Ryan Breidenbach.Spring in Action[M].北京：人民邮电出版社，2006. 04：66-87

[7] Dave Crane , Eric Pascarello , Darren James.Ajax 实战[M].北京：人民邮电出版社，2006. 11：103-127

[8] 孙印杰. Java 编程案例精解[M]. 北京：电子工业出版社, 2009. 05：78-82

[9] Marshall Lamb. Generate dynamic XML using JavaServer Pages technology [J]. http://www.ibm.com/developerworks/library/j-dynxml.html?S_TACT=105AGX52&S_CMP=cn-a-j .Dec, 2000

[10] 雷之宇.Java 项目开发实践——网络篇[M].北京：中国铁道出版社,2005.11：246-267