武汉纺织大学

Web应用开发课程设计

**图书馆借阅系统**

**学 院： 数学与计算机学院**

**班 级： 物联网11803**

**姓 名： 彭禹翀**

**学 号： 1804280311**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年12月15日**

目 录

[1 需求分析](#_Toc22456)

[1.1用户登陆](#_Toc6247)

[1.2主菜单](#_Toc15319)

[1.3各项功能](#_Toc14458)

[1.3.1 查看图书馆所有图书](#_Toc1897)

[1.3.2 借书](#_Toc7940)

[1.3.3 还书 1](#_Toc4651)

[1.3.4 查看借阅记录 1](#_Toc30015)

[1.3.5 查看个人信息 1](#_Toc5914)

[1.3.6 退出 1](#_Toc3637)

[2 系统设计 1](#_Toc21868)

[2.1 数据库设计 2](#_Toc13261)

[2.2 ER图 3](#_Toc13261)

[3 系统实现](#_Toc20261) 3

[3.1 项目结构](#_Toc17930) 3

[3.2 VO层类](#_Toc7838) 3

[3.2.1 Reader.java](#_Toc16083) 3

[3.2.2 Book.java](#_Toc6085) 5

[3.2.3 Log.java](#_Toc16083) 7

[3.3 DAO层类](#_Toc24038)

[3.3.1 ReaderDao.java](#_Toc16083)

[3.3.2 BookDao.java](#_Toc6085)

[3.3.3 IODao.java](#_Toc16083)

[3.4 工具包Util](#_Toc18558)

[3.4.1 DBConnect.java](#_Toc16083)

[3.5 Action层类 11](#_Toc24038)

[3.3.1 ReaderDao.java 11](#_Toc16083)

[3.3.2 BookDao.java 12](#_Toc6085)

[3.3.3 IODao.java 11](#_Toc16083)

[3.6 web.xml 1](#_Toc24038)3

[4系统测试](#_Toc7918)

[5 总结](#_Toc18618)

# 1 需求分析

设计一个图书管理系统，实现“登陆”、“借书”、“查看所有图书”、“还书”、“个人信息”、“查看借阅记录”等功能，具体要求如下：

## 1.1用户登陆

当程序运行时，显示“用户登陆”，提示用户依次输入用户名及其密码，当用户名及密码不正确时，显示提示信息“用户名或密码不正确，请重新输入”。

## 1.2主菜单

当用户名及密码正确时，进入系统主界面。

## 1.3各项功能

### 1.3.1 查看图书馆所有图书

当用户点击查看图书总表后，则在当前界面上显示所有的图书。

### 1.3.2 借书

当用户点击借书菜单时，输入学号和所借图书的ID号，并且选择借阅时间，在提交后即可成功借阅图书。当用户所借阅的图书数量为0时，则会提示用户该书已经借完。

### 1.3.3 还书

当用户点击还书菜单时，通过输入用户id号即可调转到还书界面，点击“归还”按钮则还书成功。

### 1.3.4 查看借阅记录

当用户点击借阅记录时则在当前界面中输出该用户的借阅信息。

### 1.3.5 查看个人信息

用户点击个人信息时即可查看自己的个人信息。

### 1.3.6 退出

用户点击退出按钮后，则立刻返回至登录界面。

# 2 系统设计

## 2.1 数据库设计

总共有3张表，分别是读者表、图书表和借书还书表。

读者表

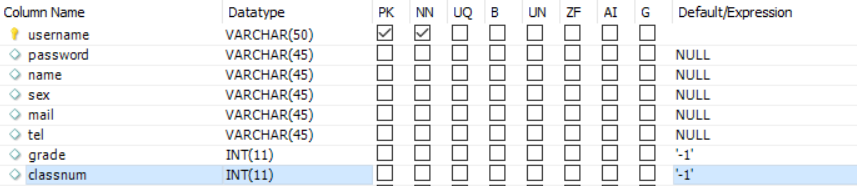


图2-1读者表

图书表如图2-2

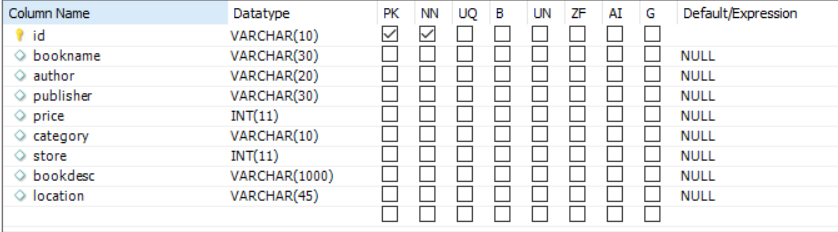


图2-2图书表

借书还书表如图2-3

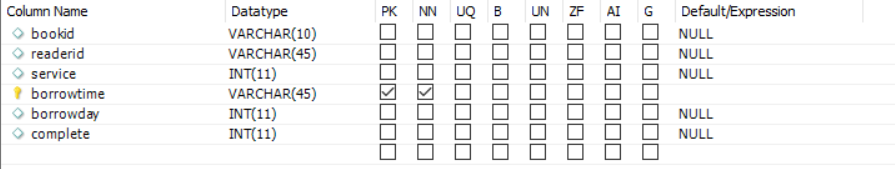


图2-3借书还书表

## 2.2 ER图

数据库设计3张表，读者表，图书表，借书还书表，对应的ER图如图3-1所示。

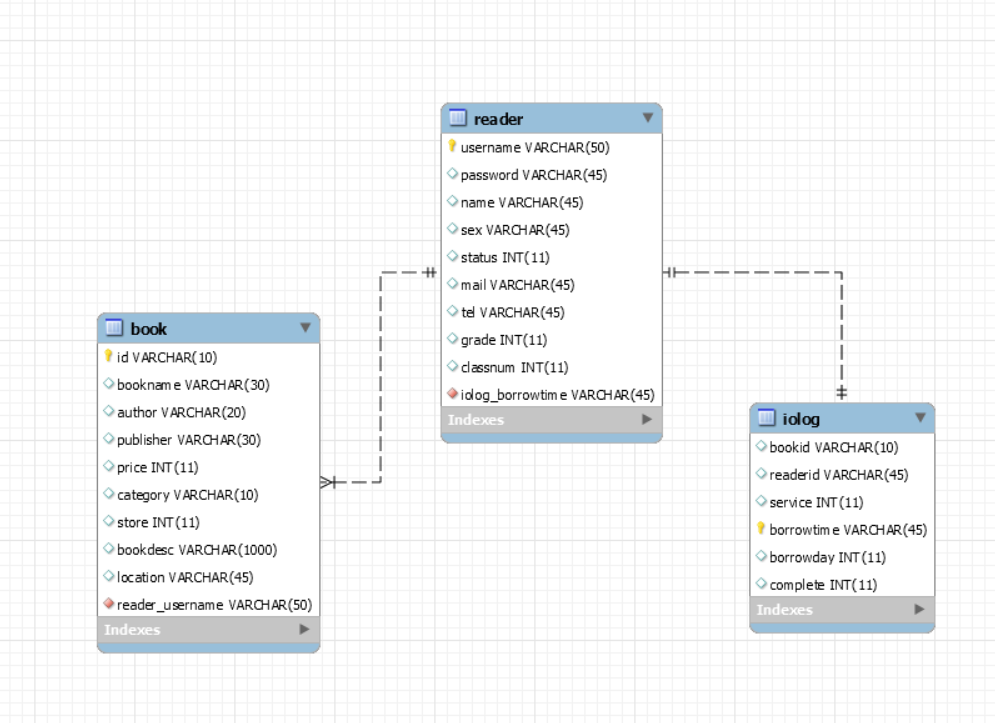
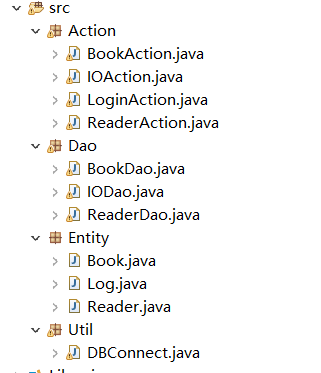


图2-4 数据库ER图

# 3 系统实现

## 3.1 项目结构



## 3.2 VO类

### 3.2.1 Reader.java

public class Reader {

private String username;

private String password;

private String name;

private String sex;

private String mail;

private int grade;

private String tel;

private int classnum;

private int borrow;

}

### 3.2.2 Book.java

public class Book {

private String id;

private String name;

private String author;

private String publisher;

private int price;

private String category;

private int store;

private int lend;

private int remain;

private String location;

private String desc;

}

### 3.2.3 Log.java

public class Log {

private String readerid;

private String service;

private String bookid;

private String borrowtime;

private int borrowday;

}

## 3.3 DAO层类

### 3.3.1 ReaderDao.java

public class ReaderDao extends DBConnect{

**QueryReaderById(String id)方法是查找读者id号**

public Reader QueryReaderById(String id){

Reader reader = new Reader();

IODao ioDao = new IODao();

Connection conn = null;

try {

conn = super.getConnection();

String sql = "SELECT \* FROM reader WHERE username=" + "'" + id + "'";

PreparedStatement pst = null;

ResultSet rs = null;

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

while (rs.next()){

reader.setName(rs.getString("name"));

reader.setSex(rs.getString("sex"));

reader.setStatus(rs.getInt("status"));

reader.setGrade(rs.getInt("grade"));

reader.setClassnum(rs.getInt("classnum"));

reader.setBorrow(ioDao.QueryBorrowNumByReaderid(id).size());

}

return reader;

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

**Login()方法是读者登录时通过账号查找用户密码和用户名。**

public String login(Reader r){

try {

Connection conn = super.getConnection();

String sql = "SELECT password,username FROM Reader WHERE username=?";

PreparedStatement pst = null;

ResultSet rs = null;

pst = conn.prepareStatement(sql);

pst.setString(1, r.getUsername());

rs = pst.executeQuery();

if (rs.next()) {

if (r.getPassword().equals(rs.getString("password"))){

r.setName(rs.getString("username"));

return "true";

}

**GetReader()方法是列出读者个人的信息。**

public ArrayList<Reader> GetReader(){

ArrayList<Reader> ReaderList = new ArrayList<>();

Connection conn = null;

try {

conn = super.getConnection();

String sql = "SELECT \* FROM reader";

PreparedStatement pst = null;

ResultSet rs = null;

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

Reader reader = null;

while(rs.next()){

reader = new Reader();

reader.setUsername(rs.getString("username"));

reader.setName(rs.getString("name"));

reader.setSex(rs.getString("sex"));

reader.setStatus(rs.getInt("status"));

reader.setMail(rs.getString("mail"));

reader.setGrade(rs.getInt("grade"));

reader.setClassnum(rs.getInt("classnum"));

reader.setTel(rs.getString("tel"));

ReaderList.add(reader);

}

return ReaderList;

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

}

### 3.3.2 BookDao.java

**getAllBook()方法主要作用是查询数据库中所有图书信息。**

public ArrayList<Book> getAllBook(){

ArrayList<Book> booklist = new ArrayList<>();

IODao ioDao = new IODao();

try {

Connection conn = super.getConnection();

String sql = "SELECT \* FROM Book";

PreparedStatement pst = null;

ResultSet rs = null;

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

while (rs.next()) {

Book book = new Book();

book.setId(rs.getString("id"));

book.setName(rs.getString("bookname"));

book.setAuthor(rs.getString("author"));

book.setPublisher(rs.getString("publisher"));

book.setPrice(rs.getInt("price"));

book.setCategory(rs.getString("category"));

book.setStore(rs.getInt("store"));

book.setLend(ioDao.QueryBookNumById(book.getId()));

book.setRemain(book.getStore() - book.getLend());

book.setLocation(rs.getString("location"));

booklist.add(book);

}

return booklist;

} catch (Exception e) {

e.printStackTrace();

}

return booklist;

}

**QueryBookById()方法是通过图书id查找图书。**

public Book QueryBookById(String s){

try {

IODao ioDao = new IODao();

Connection conn = super.getConnection();

String sql = "SELECT \* FROM Book WHERE id=" + "'" + s + "'";

PreparedStatement pst = null;

ResultSet rs = null;

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

Book book = new Book();

while (rs.next()) {

book.setId(rs.getString("id"));

book.setName(rs.getString("bookname"));

book.setAuthor(rs.getString("author"));

book.setPublisher(rs.getString("publisher"));

book.setPrice(rs.getInt("price"));

book.setCategory(rs.getString("category"));

book.setStore(rs.getInt("store"));

book.setLend(ioDao.QueryBookNumById(book.getId()));

book.setRemain(book.getStore() - book.getLend());

book.setDesc(rs.getString("bookdesc"));

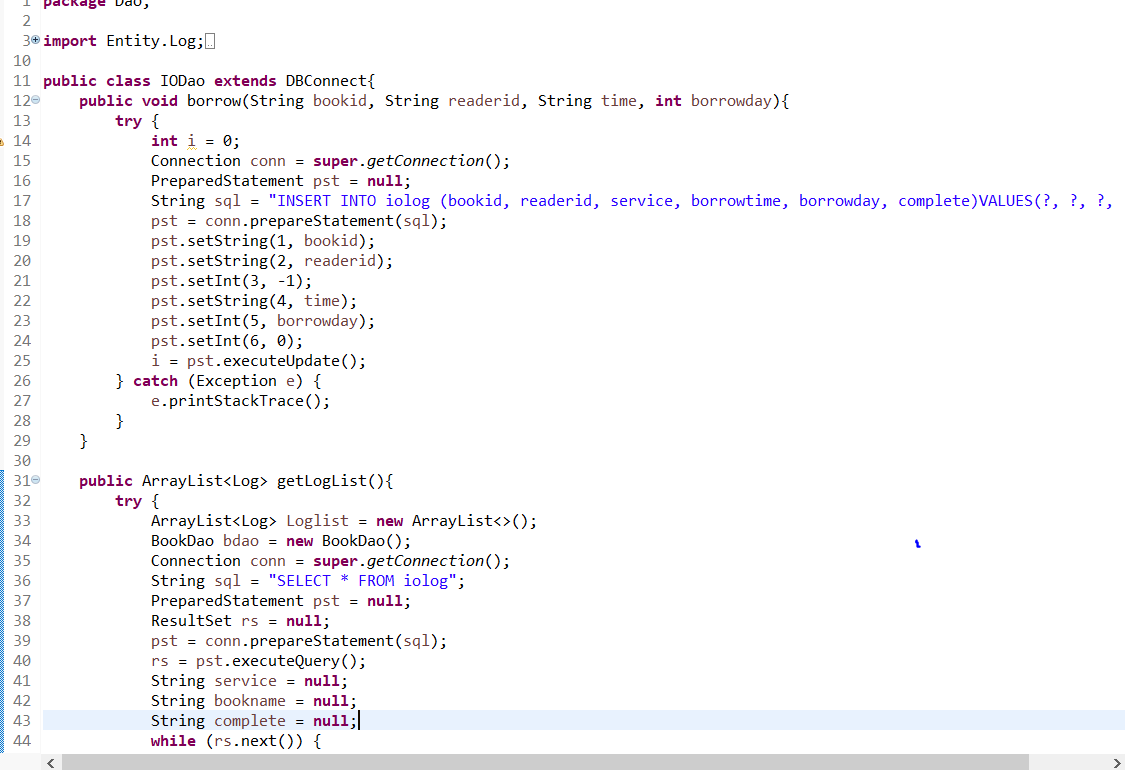
book.setLocation(rs.getString("location"));

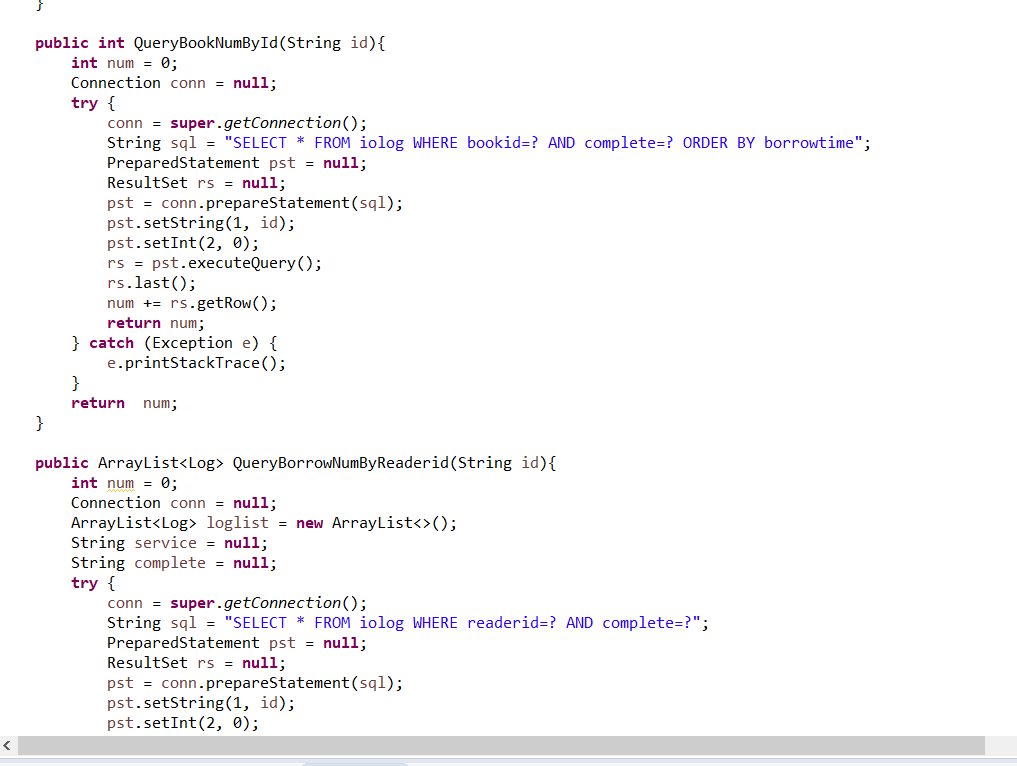
}

return book;

}

### 3.3.3 IODao.java





## 3.4 工具包Util

### 3.4.1 DBConnet.java

包内只有只有一个类，该类的作用是封装数据库，使其成为一个类，当dao层连接数据库时先调用DBConnet.java类，之后再进行其他操作。简化dao层的操作。

## 3.5 Action层

### 3.5.1 BookAction.java

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setCharacterEncoding("UTF-8");

String action = request.getParameter("action");

**if**(action.equals("getall")){

**this**.getAll(request, response);

}

**else** **if**(action.equals("querybookbyid")){

**this**.QueryBookById(request, response);

}

}

**private** **void** getAll(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

BookDao bdao = **new** BookDao();

ArrayList<Book> bookArrayList = bdao.getAllBook();

HttpSession session = request.getSession();

session.setAttribute("allbooklist", bookArrayList);

request.getRequestDispatcher("/booklist.jsp").forward(request, response);

}

**private** **void** QueryBookById(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{

BookDao bdao = **new** BookDao();

String id = request.getParameter("id");

String next = request.getParameter("next");

Book book = bdao.QueryBookById(id);

HttpSession session = request.getSession();

session.setAttribute("resultbook", book);

PrintWriter out = response.getWriter();

**if**(next.equals("check"))

request.getRequestDispatcher("/detail.jsp").forward(request, response);

**else** **if**(next.equals("edit"))

request.getRequestDispatcher("/edit.jsp").forward(request, response);

**else** **if**(next.equals("borrowcheck")){

out.write(book.getName() + "||" + book.getAuthor() + "||" + book.getPublisher() + "||" + book.getRemain());

}

**else** **if**(next.equals("existcheck")){

out.write(book.getName());

}

}

### 3.5.2 IOAction.java

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

String action = request.getParameter("action");

**if**(action.equals("borrow")){

**this**.borrow(request, response);

}

**else** **if**(action.equals("getlog")){

**this**.getlog(request, response);

}

**else** **if**(action.equals("return")){

**this**.ReturnBook(request, response);

}

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

**this**.doPost(request, response);

}

**protected** **void** borrow(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{

IODao ioDao = **new** IODao();

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy年MM月dd日 HH时mm分ss秒");

Date date = **new** Date();

String time = sdf.format(date);

String readerid = request.getParameter("readerid");

String bookid = request.getParameter("bookid");

**int** borrowday = Integer.*parseInt*(request.getParameter("borrowday"));

ioDao.borrow(bookid, readerid, time, borrowday);

**this**.getlog(request,response);

}

**protected** **void** getlog(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{

IODao ioDao = **new** IODao();

ArrayList<Log> loglist = ioDao.getLogList();

HttpSession session = request.getSession();

session.setAttribute("loglist", loglist);

request.getRequestDispatcher("/IOLog.jsp").forward(request, response);

}

**protected** **void** ReturnBook(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{

IODao ioDao = **new** IODao();

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy年MM月dd日 HH时mm分ss秒");

Date date = **new** Date();

String Returntime = sdf.format(date);

String bookid = request.getParameter("bookid");

String readerId = request.getParameter("ReaderId");

String borrowtime = request.getParameter("borrowtime");

ioDao.ReturnBook(bookid, readerId, borrowtime, Returntime);

**this**.getlog(request,response);

}

### 3.5.3 LoginAction.java

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

String action = request.getParameter("action");

**if** (action.equals("login")) {

**this**.login(request, response);

}

**else** **if**(action.equals("logout")){

**this**.logout(request, response);

}

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

doPost(request, response);

}

**private** **void** login(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

String username = **null**;

String password = **null**;

PrintWriter out = response.getWriter();

HttpSession session = request.getSession();

username = request.getParameter("username");

password = request.getParameter("password");

Reader reader=**new** Reader();

reader.setUsername(username);

reader.setPassword(password);

String result = rdao.login(reader);

**if**(result.equals("true")){

session.setAttribute("adminname", reader.getName());

request.getRequestDispatcher("/main.jsp").forward(request, response);

}

**else**{

out.write(result);

}

}

**private** **void** logout(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{

HttpSession session = request.getSession();

session.setMaxInactiveInterval(1);

session.setAttribute("logout", "1");

response.sendRedirect("/libary\_reader/index.jsp");

}

### 3.5.4 ReaderAction.java

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

String action = request.getParameter("action");

request.setCharacterEncoding("UTF-8");

response.setCharacterEncoding("UTF-8");

response.setContentType("text/html; charset=utf-8");

**if**(action.equals("QueryReaderById")) {

**this**.QueryReaderById(request, response);

}

**if** (action.equals("GetBorrowListById")){

**this**.GetBorrowListById(request, response);

}

**else** **if** (action.equals("GetAllReader")){

**this**.GetReader(request, response);

}

}

**protected** **void** QueryReaderById(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{

String readerid = request.getParameter("readerid");

ReaderDao readerDao = **new** ReaderDao();

Reader reader = readerDao.QueryReaderById(readerid);

String stauts = (reader.getStatus() == 1) ? "正常" : "异常";

String callback = reader.getName() + "||" + reader.getGrade() + "年级" + reader.getClassnum() + "班||" + stauts + "||" + reader.getBorrow(); PrintWriter out = response.getWriter();

out.write(callback);

}

**protected** **void** GetBorrowListById(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{

String id = request.getParameter("readerid");

IODao ioDao = **new** IODao();

ArrayList<Log> loglist = ioDao.QueryBorrowNumByReaderid(id);

HttpSession session = request.getSession();

session.setAttribute("loglist", loglist);

request.getRequestDispatcher("/borrowlist.jsp").forward(request,response);

}

**protected** **void** GetReader(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException{

ReaderDao readerDao = **new** ReaderDao();

ArrayList<Reader> readerlist = readerDao.GetReader();

HttpSession session = request.getSession();

session.setAttribute("readerlist", readerlist);

request.getRequestDispatcher("/ReaderList.jsp").forward(request, response);

}

## 3.6 web.xml

<servlet>

<servlet-name>LoginAction</servlet-name>

<servlet-class>Action.LoginAction</servlet-class>

</servlet>

<servlet>

<servlet-name>BookAction</servlet-name>

<servlet-class>Action.BookAction</servlet-class>

</servlet>

<servlet>

<servlet-name>IOAction</servlet-name>

<servlet-class>Action.IOAction</servlet-class>

</servlet>

<servlet>

<servlet-name>ReaderAction</servlet-name>

<servlet-class>Action.ReaderAction</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ReaderAction</servlet-name>

<url-pattern>/ReaderAction</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>LoginAction</servlet-name>

<url-pattern>/LoginAction</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>BookAction</servlet-name>

<url-pattern>/BookAction</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>IOAction</servlet-name>

<url-pattern>/IOAction</url-pattern>

</servlet-mapping>

# 4 系统测试











# 5 总结

这次课程设计我做的是图书馆借阅系统，主要是服务于读者在图书馆进行借阅书籍。系统中采用的技术主要有：jdbc数据库连接技术、mvc三层架构、js、css、jquery等一些技术，利用eclipse集成开发环境。虽然系统的基本功能差不多能实现，但还是存在系统不稳定等多个问题尚待解决。这个系统主要是我自己开发的，但也得到了老师和同学的很大帮助。我正在做系统的过程中遇到了很多问题，有的是知识存储不足，有的是考虑不够周全，之所以能够顺利实现基本功功能，离不开老师和同学的大力相助。开发一套系统，最重要的是细心，并不是一定要做到面面俱到，但也要充分考虑到客户的需求和现实意义，不管什么系统，只用运用到实际应用中，才具有先现实意义。所以在准备工作中要正确分析社会需求了解现实应用，画出流程图，把大体框架做好，然后再逐一细化。可能做不到到面面俱到，但一定要做到步步扎实，作为一个程序编程人员，要保持清醒的头脑，以现实为依据，让自己的每一行代码都能实现自己的意义。