

# 图书馆管理系统

## (JSP+Servlet+JavaBean 实现)

随着网络技术的高速发展，计算机应用的普及，利用计算机对图书馆的日常工作进行管理势在必行。虽然目前很多大型的图书馆已经有一整套比较完善的管理系统，但是在一些中小型的图书馆中，大部分工作仍需由手工完成，工作起来效率比较低，管理员不能及时了解图书馆内各类图书的借阅情况，读者需要的图书难以在短时间内找到，不便于动态及时地调整图书结构。为了更好地适应当前读者的借阅需求，解决手工管理中存在的许多弊端，越来越多的中小型图书馆正在逐步向计算机信息化管理转变。通过本章的学习，你将学到：

- ▶▶ 掌握如何做需求分析
- ▶▶ 掌握 JSP 经典设计模式中 Model2 的开发流程
- ▶▶ 掌握通过配置过滤器解决中文乱码
- ▶▶ 掌握图书馆管理系统的开发流程
- ▶▶ 掌握实现安全登录系统并防止非法用户登录的方法

## B.1 开发背景

×××图书馆是吉林省一家私营的中型图书馆企业。图书馆本着以“读者为上帝”、“为读者节省每一分钱”的服务宗旨，企业利润逐年提高，规模不断壮大、经营图书品种、数量也逐渐增多。在企业不断发展的同时，企业传统的人工方式管理暴露了一些问题。例如，读者想要借阅一本书，图书管理人员需要花费大量时间在茫茫的书海中苦苦“寻觅”，如果找到了读者想要借阅的图书还好，否则只能向读者苦笑着说“抱歉”了。企业为提高工作效率，同时摆脱图书管理人员在工作中出现的尴尬局面，现需要委托其他单位开发一个图书馆管理系统。

## B.2 系统分析

### B.2.1 需求分析

长期以来，人们使用传统的人工方式管理图书馆的日常业务，其操作流程比较繁琐。在借书时，读者首先将要借的书和借阅证交给工作人员，工作人员然后将每本书的信息卡片和读者的借阅证放在一个小栅栏里，最后在借阅证和每本书贴的借阅条上填写借阅信息。在还书时，读者首先将要还的书交给工作人员，工作人员根据图书信息找到相应的书卡和借阅证，并填好相应的还书信息。

从上述描述中可以发现，传统的手工流程存在的不足。首先处理借书、还书业务流程的效率很低；其次处理能力比较低，一段时间内，所能服务的读者人数是有限的。为此，图书馆管理系统需要为企业解决上述问题，为企业提供快速的图书信息检索功能、快捷的图书借阅和归还流程。

### B.2.2 可行性研究

根据《GB8567-88 计算机软件产品开发文件编制指南》中可行性分析的要求，制定可行性研究报告如下。

#### 1. 引言

##### 编写目的

为了给企业的决策层提供是否进行项目实施的参考依据，现以文件的形式分析项目的风险、项目需要的投资与效益。

##### 背景

×××图书馆是吉林省一家中型的私营企业。企业为了进行信息化管理、提高工作效率，现需要委托其他公司开发一个信息管理系统，项目名称为图书馆管理系统。

#### 2. 可行性研究的前提

##### 要求

图书馆管理系统要求能够提供新书登记、图书借阅、图书归还、图书借阅查询等功能。

##### 目标

图书馆管理系统的主要目标是简化图书借阅、归还的操作流程，提高员工的工作效率。

##### 条件、假定和限制

项目需要在两个月内交付用户使用。系统分析人员需要两天内到位，用户需要 5 天时间确认需求

分析文档。去除其中可能出现的问题，例如用户可能临时有事，占用 7 天时间确认需求分析。那么程序开发人员需要在 1 个月零 20 几天的时间内进行系统设计、程序编码、系统测试、程序调试和网站部署工作。其间，还包括了员工每周的休息时间。

#### 评价尺度

根据用户的要求，系统应以图书借阅和归还功能为主，对于图书的借阅和归还信息应能及时准确地保存。由于用户存在多个营业点，系统应具有局域网操作的能力，在多个营业点同时运行系统时，系统中各项操作的延时不能超过 10 秒钟。此外，在系统出现故障时，应能够及时进行恢复。

### 3. 投资及效益分析

#### 支出

根据系统的规模及项目的开发周期（两个月），公司决定投入 6 个人。为此，公司将直接支付 8 万元的工资及各种福利待遇。在项目安装及调试阶段，用户培训、员工出差等费用支出需要 1.5 万元。在项目维护阶段预计需要投入 2 万元的资金。累计项目投入需要 11.5 万元资金。

#### 收益

用户提供项目资金 25 万元。对于项目运行后进行的改动，采取协商的原则根据改动规模额外提供资金。因此从投资与收益的效益比上，公司可以获得 13.5 万元的利润。

项目完成后，会给公司提供资源储备，包括技术、经验的积累，其后再开发类似的项目时，可以极大的缩短项目开发周期。

### 4. 结论

根据上面的分析，在技术上不会存在问题，因此项目延期的可能性很小。在效益上公司投入 6 个人、两个月的时间，获利 13.5 万元，比较可观。在公司今后发展上可以储备网站开发的经验和资源。因此认为该项目可以开发。

## B.3 系统设计

### B.3.1 系统目标

根据前面所作的需求分析及用户的需求可以得出，图书馆管理系统实施后，应达到以下目标：

界面设计友好、美观。

数据存储安全、可靠。

信息分类清晰、准确。

强大的查询功能，保证数据查询的灵活性。

实现对图书借阅、续借和归还过程的全程数据信息跟踪。

提供图书借阅排行榜，为图书馆管理员提供了真实的数据信息。

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。

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。

具有易维护性和易操作性。

### B.3.2 系统功能结构

根据图书馆管理系统的特点，可以将其分为系统设置、读者管理、图书管理、图书借还、系统查

询等 5 个部分，其中各个部分及其包括的具体功能模块如图 B.1 所示。

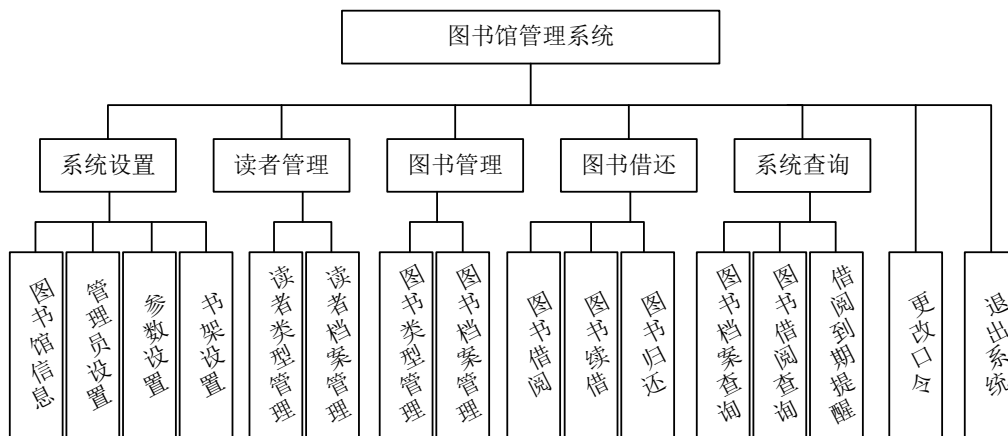


图 B.1 系统功能结构图

### B.3.3 系统流程图

图书馆管理系统的系统流程如图 B.2 所示。

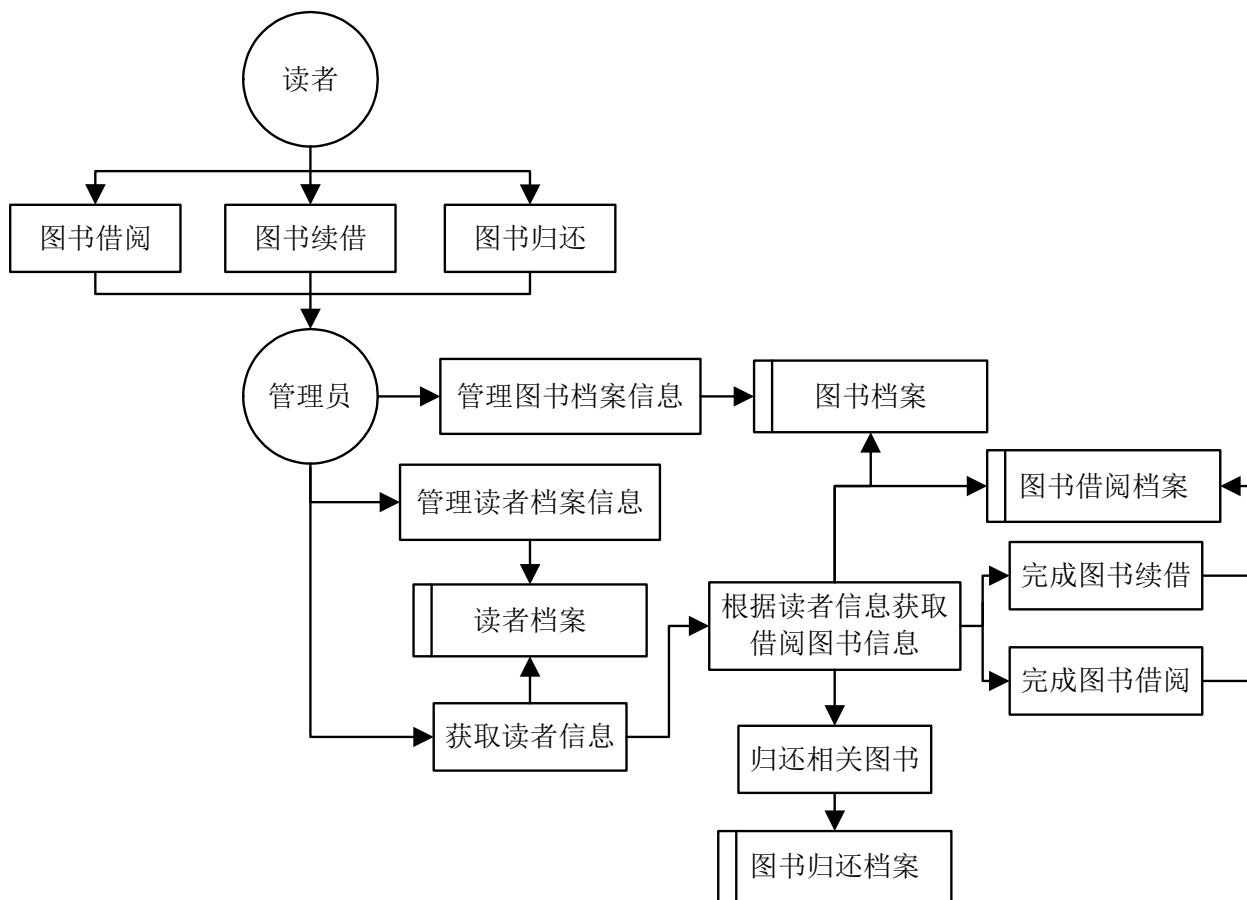


图 B.2 系统流程图

### B.3.4 开发环境

在开发图书馆管理系统时，需要具备下面的软件环境。

服务器端：

操作系统：Windows 7/Windows 2003

Web 服务器：Tomcat 6.0

Java 开发包：JDK 1.6 以上

数据库：MySQL

浏览器 IE6.0

分辨率：最佳效果为 1024×768 像素

客户端：

浏览器 IE6.0

分辨率：最佳效果为 1024×768 像素

### B.3.5 文件夹组织结构

在编写代码之前，可以把系统中可能用到的文件夹先创建出来（例如，创建一个名为 Images 的文件夹，用于保存网站中所使用的图片），这样不但可以方便以后的开发工作，还可以规范网站的整体架构。本书在开发图书馆管理系统时，设计了如图 B.3 所示的文件夹架构图。在开发时，只需要将所创建的文件保存在相应的文件夹中就可以了。

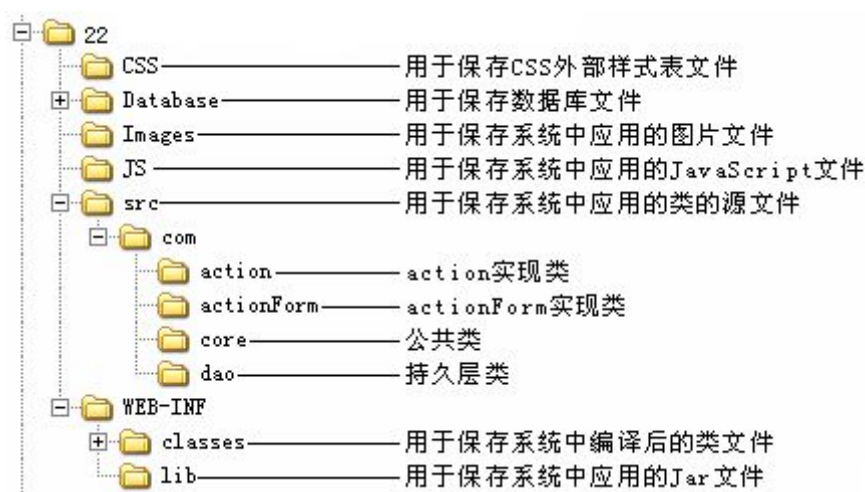


图 B.3 图书馆管理系统文件夹组织结构

## B.4 系统预览

图书馆管理系统由多个程序页面组成，下面仅列出几个典型页面。

系统登录页面如图 B.4 所示，该页面用于实现管理员登录；主界面如图 B.5 所示，该页面用于实现显示系统导航、图书借阅排行榜和版权信息等功能。



图 B.4 系统登录页面



图 B.5 主界面

图书借阅页面如图 B.6 所示，该页面用于实现图书借阅功能；图书借阅查询页面如图 B.7 所示，该页面用于实现按照符合条件查询图书借阅信息的功能。



图 B.6 图书借阅页面



图 B.7 借阅查询页面

## B.5 数据库设计

### B.5.1 数据库分析

由于本系统是为中小型图书馆开发的程序，需要充分考虑到成本问题及用户需求（如跨平台）等问题，而 MySQL 是目前最为流行的开放源码的数据库，是完全网络化的跨平台的关系型数据库系统，这正好满足了中小型企业的需求，所以本系统采用 MySQL 数据库。

### B.5.2 数据库概念设计

根据以上各节对系统所做的需求分析和系统设计，规划出本系统中使用的数据库实体分别为图书档案实体、读者档案实体、图书借阅实体、图书归还实体和管理员实体。下面将介绍几个关键实体的 E-R 图。

#### 图书档案实体

图书档案实体包括编号、条形码、书名、类型、作者、译者、出版社、价格、页码、书架、库存总量、录入时间、操作员和是否删除等属性。其中“是否删除属性”用于标记图书是否被删除，由于图

书馆中的图书信息不可以被随意删除，所以即使当某种图书不能再借阅，而需要删除其档案信息时，也只能采用设置删除标记的方法。图书档案实体的 E-R 图如图 B.8 所示。

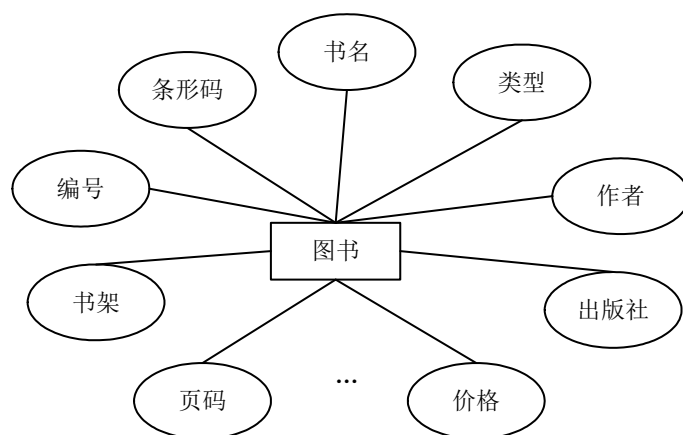


图 B.8 图书档案实体 E-R 图

#### 读者档案实体

读者档案实体包括编号、姓名、性别、条形码、职业、出生日期、有效证件、证件号码、电话、电子邮件、登记日期、操作员、类型和备注等属性。读者档案实体的 E-R 图如图 B.9 所示。

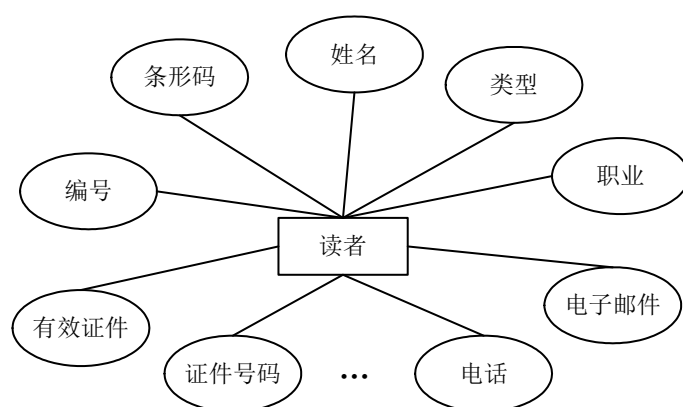


图 B.9 读者档案实体 E-R 图

#### 借阅档案实体

借阅档案实体包括编号、读者编号、图书编号、借书时间、应还时间、操作员和是否归还等属性。借阅档案实体的 E-R 图如图 B.10 所示。

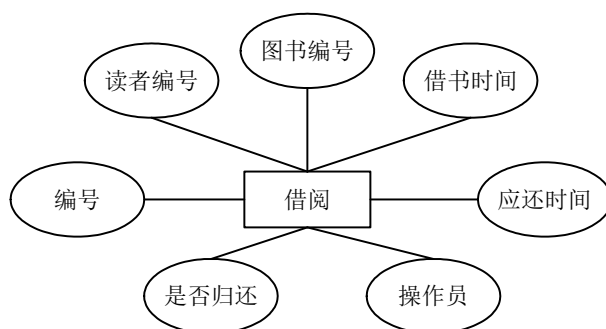


图 B.10 借阅档案实体 E-R 图

归还档案实体

归还档案实体包括编号、读者编号、图书编号、归还时间和操作员等属性。借阅档案实体的 E-R 图如图 B.11 所示。

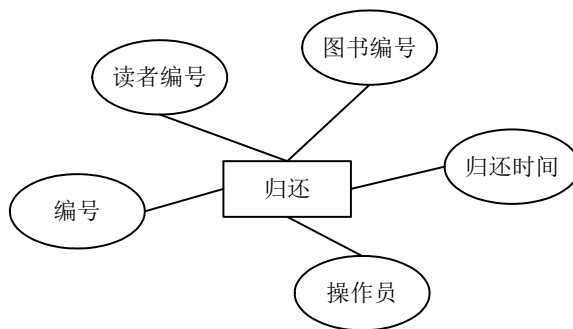


图 B.11 归还档案实体 E-R 图

### B.5.3 数据库逻辑结构

在数据库概念设计中已经分析了本系统中主要的数据实体对象，通过这些实体可以得出数据表结构的基本模型，最终实施到数据库中，形成完整的数据结构。为了使读者对本系统的数据库的结构有一个更清晰的认识，下面给出数据库中所包含的数据表的结构图，如图 B.12 所示。

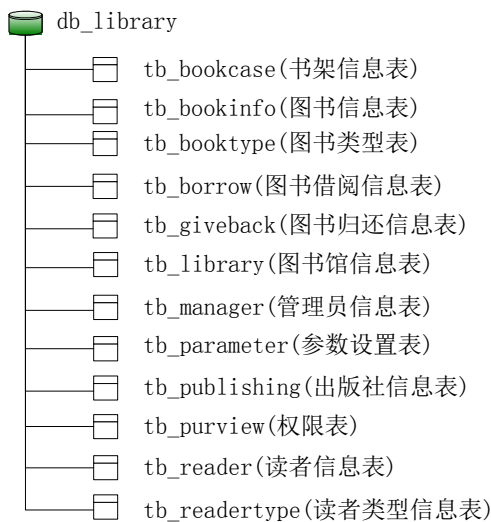


图 B.12 db\_library 数据库所包含数据表的结构图

本系统共包含 12 张数据表，限于篇幅，这里只给出比较重要的数据表。

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。表 tb\_manager 的结构如表 B.1 所示。

表 B.1

表 tb\_manager 的结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	int(10)unsigned	No	Yes		ID(自动编号)
name	varchar(30)	Yes		NULL	管理员名称
pwd	varchar(30)	Yes		NULL	密码



## tb\_purview (权限表)

权限表主要用来保存管理员的权限信息, 该表中的 id 字段与管理员信息表 (tb\_manager) 中的 id 字段相关联。表 tb\_purview 的结构如表 B.2 所示。

表 B.2 表 tb\_purview 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	描 述
id	int(11)	No	Yes	0	管理员ID号
sysset	tinyint(1)	Yes		0	系统设置
readerset	tinyint(1)	Yes		0	读者管理
bookset	tinyint(1)	Yes		0	图书管理
borrowback	tinyint(1)	Yes		0	图书借还
sysquery	tinyint(1)	Yes		0	系统查询

## tb\_bookinfo (图书信息表)

图书信息表主要用来保存图书信息。表 tb\_bookinfo 的结构如表 B.3 所示。

表 B.3 表 tb\_bookinfo 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	描 述
barcode	varchar(30)	Yes		NULL	条形码
bookname	varchar(70)	Yes		NULL	书名
typeid	int(10)unsigned	Yes		NULL	类型
author	varchar(30)	Yes		NULL	作者
translator	varchar(30)	Yes		NULL	译者
ISBN	varchar(20)	Yes		NULL	出版社
price	float(8,2)	Yes		NULL	价格
page	int(10)unsigned	Yes		NULL	页码
bookcase	int(10)unsigned	Yes		NULL	书架
inTime	date	Yes		NULL	录入时间
operator	varchar(30)	Yes		NULL	操作员
del	tinyint(1)	Yes		0	是否删除
id	int(11)	No	Yes		ID(自动编号)

## tb\_parameter (参数设置表)

参数设置表主要用来保存办证费及书证的有效期限等信息。表 tb\_parameter 的结构如表 B.4 所示。

表 B.4 表 tb\_parameter 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	描 述
id	int(10)unsigned	No	Yes		ID(自动编号)
cost	int(10)unsigned	Yes		NULL	办证费
validity	int(10)unsigned	Yes		NULL	有效期限

## tb\_booktype (图书类型表)

图书类型表主要用来保存图书类型信息。表 tb\_booktype 的结构如表 B.5 所示。

表 B.5 表 tb\_booktype 的结构

字 段 名	数 据 类 型	是 否 为 空	是 否 主 键	默 认 值	描 述
id	int(10)unsigned	No	Yes		ID(自动编号)
typename	varchar(30)	Yes		NULL	类型名称
days	int(10)unsigned	Yes		NULL	可借天数

tb\_bookcase（书架信息表）

书架信息表主要用来保存书架信息。表 tb\_bookcase 的结构如表 B.6 所示。

表 B.6 表 tb\_bookcase 的结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	int(10)unsigned	No	Yes		ID（自动编号）
name	varchar(30)	Yes		NULL	书架名称

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。表 tb\_borrow 的结构如表 B.7 所示。

表 B.7 表 tb\_borrow 的结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	int(10)unsigned	No	Yes		ID（自动编号）
readerid	int(10)unsigned	Yes		NULL	读者编号
bookid	int(10)	Yes		NULL	图书编号
borrowTime	date	Yes		NULL	借书时间
backtime	date	Yes		NULL	应还时间
operator	varchar(30)	Yes		NULL	操作员
ifback	tinyint(1)	Yes		0	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。表 tb\_giveback 的结构如表 B.8 所示。

表 B.8 表 tb\_giveback 的结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	int(10)unsigned	No	Yes		ID（自动编号）
readerid	int(11)	Yes		NULL	读者编号
bookid	int(11)	Yes		NULL	图书编号
backTime	date	Yes		NULL	归还时间
operator	varchar(30)	Yes		NULL	操作员

tb\_readertype（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。表 tb\_readertype 的结构如表 B.9 所示。

表 B.9 表 tb\_readertype 的结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	int(10) unsigned	No	Yes		ID（自动编号）
name	varchar(50)	Yes		NULL	名称
number	int(4)	Yes		NULL	可借数量

tb\_reader（读者信息表）

读者信息表主要用来保存读者信息。表 tb\_reader 的结构如表 B.10 所示。

表 B.10 表 tb\_reader 的结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	int(10) unsigned	No	Yes		ID（自动编号）
name	varchar(20)	Yes		NULL	姓名
sex	varchar(4)	Yes		NULL	性别
barcode	varchar(30)	Yes		NULL	条形码
vocation	varchar(50)	Yes		NULL	职业
birthday	date	Yes		NULL	出生日期

paperType	varchar(10)	Yes		NULL	有效证件
paperNO	varchar(20)	Yes		NULL	证件号码
tel	varchar(20)	Yes		NULL	电话
email	varchar(100)	Yes		NULL	电子邮件
createDate	date	Yes		NULL	登记日期
operator	varchar(30)	Yes		NULL	操作员
remark	text	Yes		NULL	备注
typeid	int(11)	Yes		NULL	类型

## B.6 公共模块设计

在开发过程中，经常会用到一些公共模块，例如，数据库连接及操作的类、字符串处理的类及解决中文乱码的过滤器等，因此，在开发系统前首先需要设计这些公共模块。下面将具体介绍图书馆管理系统中所需要的公共模块的设计过程。

### B.6.1 数据库连接及操作类的编写

数据库连接及操作类通常包括连接数据库的方法 `getConnection()`、执行查询语句的方法 `executeQuery()`、执行更新操作的方法 `executeUpdate()`、关闭数据库连接的方法 `close()`。下面将详细介绍如何编写图书馆管理系统中的数据库连接及操作的类 `ConnDB`。

(1) 指定类 `ConnDB` 保存的包，并导入所需的类包，本例将其保存到 `com.core` 包中，代码如下：

```
package com.core;           //将该类保存到com.core包中
import java.io.InputStream; //导入java.io.InputStream类
import java.sql.*;          //导入java.sql包中的所有类
import java.util.Properties; //导入java.util.Properties类
```

注意：包语句以关键字 `package` 后面紧跟一个包名称，然后以分号“;”结束；包语句必须出现在 `import` 语句之前；一个 `.java` 文件只能有一个包语句。

(2) 定义 `ConnDB` 类，并定义该类中所需的全局变量及构造方法，代码如下：

```
public class ConnDB {
    public Connection conn = null;           //声明Connection对象的实例
    public Statement stmt = null;           //声明Statement对象的实例
    public ResultSet rs = null;             //声明ResultSet对象的实例
    private static String propFileName = "/com/connDB.properties"; //指定资源文件保存的位置
    private static Properties prop = new Properties(); //创建并实例化Properties对象的实例
    private static String dbClassName = "com.mysql.jdbc.Driver"; //定义保存数据库驱动名的变量
    private static String dbUrl = "jdbc:mysql://127.0.0.1:3306/db_library?user=root&password=111&useUnicode=true";
    public ConnDB() {                       //构造方法
        try {                               //捕捉异常
            //将Properties文件读取到InputStream对象中
            InputStream in = getClass().getResourceAsStream(propFileName);
            prop.load(in);                  //通过输入流对象加载Properties文件
            dbClassName = prop.getProperty("DB_CLASS_NAME"); //获取数据库驱动
            //获取连接的URL
            dbUrl = prop.getProperty("DB_URL", dbUrl);
        }
    }
}
```

```

        catch (Exception e) {
            e.printStackTrace();           //输出异常信息
        }
    }
}

```

(3) 为了方便程序移植, 这里将数据库连接所需信息保存到 `properties` 文件中, 并将该文件保存在 `com` 包中。`connDB.properties` 文件的内容如下:

```

#DB_CLASS_NAME(驱动的类的类名)
DB_CLASS_NAME=com.mysql.jdbc.Driver
#DB_URL (要连接数据库的地址)
DB_URL=jdbc:mysql://127.0.0.1:3306/db_library?user=root&password=111&useUnicode=true

```

说明: `properties` 文件为本地资料文本文件, 以“消息/消息文本”的格式存放数据, 文件中“#”的后面为注释行。使用 `Properties` 对象时, 首先需创建并实例化该对象, 代码如下:

```
private static Properties prop = new Properties();
```

再通过文件输入流对象加载 `Properties` 文件, 代码如下:

```
prop.load(new FileInputStream(propFileName));
```

最后通过 `Properties` 对象的 `getProperty` 方法读取 `properties` 文件中的数据。

(4) 创建连接数据库的方法 `getConnection()`, 该方法返回 `Connection` 对象的一个实例。`getConnection()` 方法的代码如下:

```

public static Connection getConnection() {
    Connection conn = null;
    try {
        ❶ Class.forName(dbClassName).newInstance(); //装载数据库驱动
        ❷ conn = DriverManager.getConnection(dbUrl); //建立与数据库URL中定义的数据库的连接
    }
    catch (Exception ee) {
        ee.printStackTrace();           //输出异常信息
    }
    if (conn == null) {
        System.err.println(
            "警告: DbConnectionManager.getConnection() 获得数据库连接失败.\r\n\r\n链接类型:" +
            dbClassName + "\r\n链接位置:" + dbUrl); //在控制台上输出提示信息
    }
    return conn; //返回数据库连接对象
}

```

### 关键代码解析

❶ 该句代码用于利用 `Class` 类中的静态方法 `forName()`, 加载要使用的 `Driver`。使用该语句可以将传入的 `Driver` 类名称的字符串当作 `forN`。

❷ `DriverManager` 用于管理 `JDBC` 驱动程序的接口, 通过其 `getConnection()` 方法来获取 `Connection` 对象的引用。`Connection` 对象的常用方法如下。

`Statement createStatement()`: 创建一个 `Statement` 对象, 用于执行 `SQL` 语句。

`close()`: 关闭数据库的连接, 在使用完连接后必须关闭, 否则连接会保持一段比较长的时间, 直到超时。

`PreparedStatement prepareStatement(String sql)`: 使用指定的 `SQL` 语句创建了一个预处理语句, `sql` 参数中往往包含一个或多个“?”占位符。

`CallableStatement prepareCall(String sql)`: 创建一个 `CallableStatement` 用于执行存储过程, `sql` 参数是调用的存储过程,

中间至少包含一个“?”占位符。

(5) 创建执行查询语句的方法 `executeQuery`，返回值为 `ResultSet` 结果集。`executeQuery` 方法的代码如下：

```
public ResultSet executeQuery(String sql) {
    try {
        conn = getConnection();
        ① stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY);
        ② rs = stmt.executeQuery(sql);
    }
    catch (SQLException ex) {
        System.err.println(ex.getMessage());
    }
    return rs;
}
```

#### 关键代码解析

① `ResultSet.TYPE_SCROLL_INSENSITIVE` 常量允许记录指针向前或向后移动，且当 `ResultSet` 对象变动记录指针时，会影响记录指针的位置。

`ResultSet.CONCUR_READ_ONLY` 常量可以解释为 `ResultSet` 对象仅能读取，不能修改，在对数据库的查询操作中使用。

② `stmt` 为 `Statement` 对象的一个实例，通过其 `executeQuery(String sql)` 方法可以返回一个 `ResultSet` 对象。

(6) 创建执行更新操作的方法 `executeUpdate()`，返回值为 `int` 型的整数，代表更新的行数。`executeUpdate()` 方法的代码如下：

```
public int executeUpdate(String sql) {
    int result = 0;
    try {
        conn = getConnection();
        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY);
        result = stmt.executeUpdate(sql);
    } catch (SQLException ex) {
        result = 0;
    }
    return result;
}
```

(7) 创建关闭数据库连接的方法 `close()`。`close()` 方法的代码如下：

```
public void close() {
    try {
        if (rs != null) {
            rs.close();
        }
        if (stmt != null) {
            stmt.close();
        }
        if (conn != null) {
            conn.close();
        }
    }
}
```

```

    }
} catch (Exception e) {
    e.printStackTrace(System.err);           //输出异常信息
}
}

```

## B.6.2 字符串处理类的编写

字符串处理的类是解决程序中经常出现的有关字符串处理问题方法的类，本实例中只包括过滤字符串中的危险字符的方法 `filterStr()`。`filterStr()`方法的代码如下：

```

public static final String filterStr(String str){
    str=str.replaceAll(";","");           //替换字符串中的;为空
    str=str.replaceAll("&","&");         //替换字符串中的&为&
    str=str.replaceAll("<","&lt;");       //替换字符串中的<为&lt;
    str=str.replaceAll(">","&gt;");       //替换字符串中的>为&gt;
    str=str.replaceAll("'", "");          //替换字符串中的'为空
    str=str.replaceAll("--"," ");         //替换字符串中的--为空格
    str=str.replaceAll("/", "");          //替换字符串中的/为空
    str=str.replaceAll("%","");          //替换字符串中的%为空
    return str;
}

```

## B.6.3 配置解决中文乱码的过滤器

在程序开发时，通常有两种方法解决程序中经常出现的中文乱码问题，一种是通过编码字符串处理类，对需要的内容进行转码；另一种是配置过滤器。其中，第二种方法比较方便，只需要在开发程序时配置正确即可。下面将介绍本系统中配置解决中文乱码的过滤器的具体步骤。

(1)编写 `CharacterEncodingFilter` 类，让它实现 `Filter` 接口，成为一个 `Servlet` 过滤器，在实现 `doFilter()` 接口方法时，根据配置文件中设置的编码格式参数分别设置请求对象的编码格式和应答对象的内容类型参数。

```

public class CharacterEncodingFilter implements Filter {
    protected String encoding = null;           // 定义编码格式变量
    protected FilterConfig filterConfig = null; // 定义过滤器配置对象
    public void init(FilterConfig filterConfig) throws ServletException {
        this.filterConfig = filterConfig;       // 初始化过滤器配置对象
        this.encoding = filterConfig.getInitParameter("encoding"); // 获取配置文件中指定的编码格式
    }
    // 过滤器的接口方法，用于执行过滤业务
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        if (encoding != null) {
            request.setCharacterEncoding(encoding); // 设置请求的编码
            // 设置应答对象的内容类型（包括编码格式）
            response.setContentType("text/html; charset=" + encoding);
        }
        chain.doFilter(request, response);       // 传递给下一个过滤器
    }
}

```

```
public void destroy() {  
    this.encoding = null;  
    this.filterConfig = null;  
}  
}
```

(2) 在 web-inf.xml 文件中配置过滤器，并设置编码格式参数和过滤器的 URL 映射信息。关键代码如下。

```
<filter>  
    <filter-name>CharacterEncodingFilter</filter-name>  
    <filter-class>com.CharacterEncodingFilter</filter-class>    <!--指定过滤器类文件-->  
    <init-param>  
        <param-name>encoding</param-name>  
        <param-value>GBK</param-value>    <!--指定编码为GBK编码-->  
    </init-param>  
</filter>  
<filter-mapping>  
    <filter-name>CharacterEncodingFilter</filter-name>  
    <url-pattern>/*</url-pattern>  
    <!--设置过滤器对应的请求方式-->  
    <dispatcher>REQUEST</dispatcher>  
    <dispatcher>FORWARD</dispatcher>  
</filter-mapping>
```

## B.7 主界面设计

### B.7.1 主界面概述

管理员通过“系统登录”模块的验证后，可以登录到图书馆管理系统的主界面。系统主界面主要包括 Banner 信息栏、导航栏、排行榜和版权信息 4 部分。其中，导航栏中的功能菜单将根据登录管理员的权限进行显示。例如，系统管理员 mr 登录后，将拥有整个系统的全部功能，因为它是超级管理员。主界面运行结果如图 B.13 所示。



图 B.13 系统主界面的运行结果

## B.7.2 主界面技术分析

在如图 B.13 所示的主界面中，Banner 信息栏、导航栏和版权信息，并不是仅存在于主界面中，其他功能模块的子界面中也需要包括这些部分。因此，可以将这几个部分分别保存在单独的文件中，这样，在需要放置相应功能时只需包含这些文件即可，主界面的布局如图 B.14 所示。

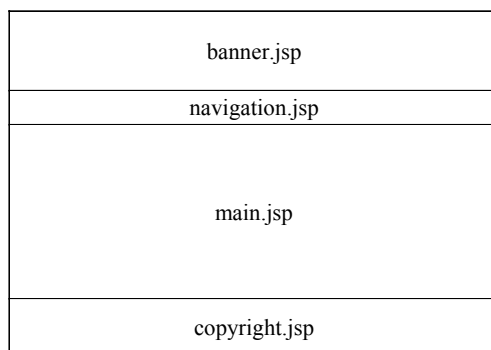


图 B.14 主界面的布局

在 JSP 页面中包含文件有两种方法：一种是应用 `<%@ include %>` 指令实现，另一种是应用 `<jsp:include>` 动作元素实现。

`<%@ include %>` 指令用来在 JSP 页面中包含另一个文件。包含的过程是静态的，即在指定文件属性值时，只能是一个包含相对路径的文件名，而不能是一个变量，也不可以在所指定的文件后面添加任何参数。其语法格式如下：

```
<%@ include file="fileName"%>
```

`<jsp:include>` 动作元素可以指定加载一个静态或动态的文件，但运行结果不同。如果指定为静态文件，那么这种指定仅仅是把指定的文件内容加到 JSP 文件中去，则这个文件不被编译。如果是动态文



件，那么这个文件将会被编译器执行。由于在页面中包含查询模块时，只需要将文件内容添加到指定的 JSP 文件中即可，所以此处可以使用加载静态文件的方法包含文件。应用<jsp:include>动作元素加载静态文件的语法格式如下：

```
<jsp:include page="{relativeURL | <%=expression%>}" flush="true"/>
```

使用<%@ include %>指令和<jsp:include>动作元素包含文件的区别是：使用<%@ include %>指令包含的页面，是在编译阶段将该页面的代码插入到了主页面的代码中，最终包含页面与被包含页面生成了一个文件。因此，如果被包含页面的内容有改动，需重新编译该文件。而使用<jsp:include>动作元素包含的页面可以是动态改变的，它是在 JSP 文件运行过程中被确定的，程序执行的是两个不同的页面，即在主页面中声明的变量，在被包含的页面中是不可见的。由此可见，当被包含的 JSP 页面中包含动态代码时，为了不和主页面中的代码相冲突，需要使用<jsp:include>动作元素包含文件。应用<jsp:include>动作元素包含查询页面的代码如下：

```
<jsp:include page="search.jsp" flush="true"/>
```

考虑到本系统中需要包含的多个文件之间相对比较独立，并且不需要进行参数传递，属于静态包含，因此采用<%@ include %>指令实现。

### B.7.3 主界面的实现过程

应用<%@ include %>指令包含文件的方法进行主界面布局的代码如下：

```
❶ <%@include file="banner.jsp"%>
❷ <%@include file="navigation.jsp"%>
❸ <!--显示图书借阅排行榜-->
   <table width="778" height="510" border="0" align="center" cellpadding="0" cellspacing="0"
bgcolor="#FFFFFF"
   class="tableBorder_gray">
       <tr>
           <td align="center" valign="top" style="padding:5px;">
               ...           <!--此处省略了显示图书借阅排行的代码-->
           </td>
       </tr>
   </table>
❹ <%@ include file="copyright.jsp"%>
```

#### 关键代码解析

- ❶ 应用<%@ include %>指令包含 banner.jsp 文件，该文件用于显示 Banner 信息及当前登录管理员。
- ❷ 应用<%@ include %>指令包含 navigation.jsp 文件，该文件用于显示当前系统时间及系统导航菜单。
- ❸ 在主界面（main.jsp）中，应用表格布局的方式显示图书借阅排行榜。
- ❹ 应用<%@ include %>指令包含 copyright.jsp 文件，该文件用于显示版权信息。

## B.8 管理员模块设计

### B.8.1 管理员模块概述

管理员模块主要包括管理员登录、查看管理员列表、添加管理员信息、管理员权限设置、管理员

删除和更改口令 6 个功能。管理员模块的框架如图 B.15 所示。

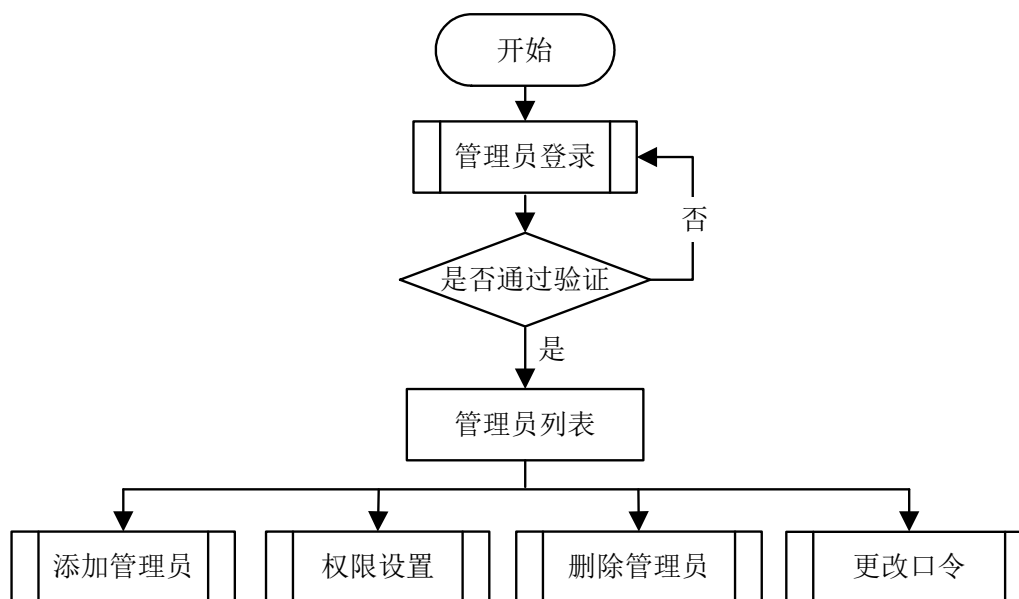


图 B.15 管理员模块的框架图

## B.8.2 管理员模块技术分析

由于本系统采用的是 JSP 经典设计模式中的 Model2，即 JSP+Servlet+JavaBean，该开发模式遵循 MVC 设计理念。所以在实现管理员模块时，需要编写管理员模块对应的实体类和 Servlet 控制类。在 MVC 中，实体类属于模型层，用于封装实体对象，是一个具有 getXXX()和 setXXX()方法的类。请求控制类属于控制层，用于接收各种业务请求，是一个 Servlet。下面将详细介绍如何编写管理员模块的实体类和 Servlet 控制类。

### 1. 编写管理员的实体类

在管理员模块中，涉及到的数据表是 tb\_manager（管理员信息表）和 tb\_purview（权限表），其中，管理员信息表中保存的是管理员名称和密码等信息，权限表中保存的是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，根据这些信息可以得出管理员模块的实体类。管理员模块的实体类的名称为 ManagerForm，具体代码如下：

```
package com.actionForm;
public class ManagerForm {
    private Integer id=new Integer(-1);           //管理员ID号
    private String name="";                       //管理员名称
    private String pwd="";                       //管理员密码
    private int sysset=0;                         //系统设置权限
    private int readerset=0;                     //读者管理权限
    private int bookset=0;                      //图书管理权限
    private int borrowback=0;                   //图书借还权限
    private int sysquery=0;                     //系统查询权限
    /*****提供控制ID属性的方法*****/
    public Integer getId() {                     //id属性的getXXX()方法
        return id;
    }
}
```

```

    }
    public void setId(Integer id) {          //id属性的setXXX()方法
        this.id = id;
    }
    /*****
    ...          //此处省略了其他控制管理员信息的getXXX()和setXXX()方法
    *****/
}

```

## 2. 编写管理员的 Servlet 控制类

管理员功能模块的 Servlet 控制类继承了 HttpServlet 类，在该类中，首先需要在构造方法中实例化管理员模块的 ManagerDAO 类（该类用于实现与数据库的交互），然后编写 doGet()和 doPost()方法，在这两个方法中根据 request 的 getParameter()方法获取的 action 参数值执行相应方法，由于这两个方法中的代码相同，所以只需在第一个方法 doGet()中写相应代码，在另一个方法 doPost()中调用 doGet()方法即可。

管理员模块的 Servlet 控制类的关键代码如下：

```

public class Manager extends HttpServlet {
    private ManagerDAO managerDAO = null;          // 声明ManagerDAO的对象
    public Manager() {
        this.managerDAO = new ManagerDAO();        // 实例化ManagerDAO类
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String action = request.getParameter("action");
        if (action == null || "".equals(action)) {
            request.getRequestDispatcher("error.jsp").forward(request, response);
        } else if ("login".equals(action)) {        // 当action值为login时，调用managerLogin()方法验证管理员身份
            managerLogin(request, response);
        } else if ("managerAdd".equals(action)) {
            managerAdd(request, response);           // 添加管理员信息
        } else if ("managerQuery".equals(action)) {
            managerQuery(request, response);         // 查询管理员及权限信息
        } else if ("managerModifyQuery".equals(action)) {
            managerModifyQuery(request, response);   // 设置管理员权限时查询管理员信息
        } else if ("managerModify".equals(action)) {
            managerModify(request, response);        // 设置管理员权限
        } else if ("managerDel".equals(action)) {
            managerDel(request, response);           // 删除管理员
        } else if ("querypwd".equals(action)) {
            pwdQuery(request, response);             // 更改口令时应用的查询
        } else if ("modifypwd".equals(action)) {
            modifypwd(request, response);            // 更改口令
        }
        public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            doGet(request, response);
        }
        ...
    }
}
//此处省略了该类中的其他方法，这些方法将在后面的具体过程中给出

```

### 3. 配置管理员的 servlet 控制类

管理员的 servlet 控制类编写完毕后，还需要在 web.xml 文件中配置该 servlet，关键代码如下：

```
<servlet>
    <servlet-name>Manager</servlet-name>
    <servlet-class>com.action.Manager</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Manager</servlet-name>
    <url-pattern>/manager</url-pattern>
</servlet-mapping>
```

## B.8.3 系统登录的实现过程

系统登录使用的数据表为：tb\_manager。

系统登录是进入图书馆管理系统的入口。在运行本系统后，首先进入的是系统登录页面，在该页面中，系统管理员可以通过输入正确的管理员名称和密码登录到系统，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，并给予提示信息。系统登录的运行结果如图 B.16 所示。



图 B.16 系统登录的运行结果

**注意：**在实现系统登录前，需要在 MySQL 数据库中，手动添加一条系统管理员的数据（管理员名为 mr，密码为 mrsoft，拥有所有权限），即在 MySQL 的客户端命令行中应用下面的语句分别向管理员信息表 tb\_manager 和权限表 tb\_purview 中各添加一条数据记录。

```
#添加管理员信息
insert into tb_manager (name,pwd) values(mr,'mrsoft');
#添加权限信息
insert into tb_purview values(1,1,1,1,1,1);
```

### 1. 设计系统登录页面

系统登录页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，该页面中所涉及到的表单元素如表 B.11 所示。

表 B.11 系统登录页面所涉及的表单元素

名 称	元 素 类 型	重 要 属 性	含 义
form1	form	method="post" action="manager?action=login"	管理员登录表单
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)"	“确定”按钮
Submit3	reset	value="重置"	“重置”按钮
Submit2	button	value="关闭" onClick="window.close();"	“关闭”按钮

编写自定义的 JavaScript 函数，用于判断管理员名称和密码是否为空。代码如下：

```
<script language="javascript">
function check(form){
    if (form.name.value==""){           //判断管理员名称是否为空
        alert("请输入管理员名称!");form.name.focus();return false;
    }
    if (form.pwd.value==""){           //判断密码是否为空
        alert("请输入密码!");form.pwd.focus();return false;
    }
}
</script>
```

## 2. 修改管理员的 servlet 控制类

在管理员登录页面的管理员名称和管理员密码文本框中输入正确的管理员名称和密码后，单击“确定”按钮，网页会访问一个 URL，这个 URL 是 manager?action=login。从该 URL 地址中可以知道系统登录模块涉及到的 action 的参数值为 login，也就是当 action=login 时，会调用验证管理员身份的方法 managerLogin()，具体代码如下：

```
if (action == null || "".equals(action)) {           //判断action的参数值是否为空
    request.getRequestDispatcher("error.jsp").forward(request, response);           //转到错误提示页
} else if ("login".equals(action)) {                 // 当action值为login时，调用managerLogin()方法验证管理员身份
    managerLogin(request, response);                 //调用验证管理员身份的方法
}
```

在验证管理员身份的方法 managerLogin()中，首先需要将接收到的表单信息保存到管理员实体类 ManagerForm 中，然后调用 ManagerDAO 类中的 checkManager()方法验证登录管理员信息是否正确，如果正确将管理员名称保存到 session 中，并将页面重定向到系统主界面，否则将错误提示信息“您输入的管理员名称或密码错误！”保存到 HttpServletRequest 的对象 error 中，并重定向页面至错误提示页。验证管理员身份的方法 managerLogin()的具体代码如下：

```
public void managerLogin(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    ManagerForm managerForm = new ManagerForm();           //实例化managerForm类
    managerForm.setName(request.getParameter("name"));       //获取管理员名称并设置name属性
    managerForm.setPwd(request.getParameter("pwd"));         //获取管理员密码并设置pwd属性
    int ret = managerDAO.checkManager(managerForm);          //调用ManagerDAO类的checkManager()方法
    if (ret == 1) {
        /*****将登录到系统的管理员名称保存到session中*****/
        HttpSession session=request.getSession();
        session.setAttribute("manager",managerForm.getName());
        /*****/
    }
```

```

        request.getRequestDispatcher("main.jsp").forward(request, response);           //转到系统主界面
    } else {
        request.setAttribute("error", "您输入的管理员名称或密码错误！");
        request.getRequestDispatcher("error.jsp").forward(request, response);       //转到错误提示页
    }
}

```

### 3. 编写系统登录的 ManagerDAO 类的方法

从 managerLogin()方法中可以知道系统登录页调用的 ManagerDAO 类的方法是 checkManager()。在 checkManager()方法中, 首先从数据表 tb\_manager 中查询输入的管理员名称是否存在, 如果存在, 再判断查询到的密码是否与输入的密码相等, 如果相等, 将标志变量设置为 1, 否则设置为 0; 反之如果不存在, 则将标志变量设置为 0。checkManager()方法的具体代码如下:

```

public int checkManager(ManagerForm managerForm) {
    int flag = 0;
    ChStr chStr=new ChStr();
    String sql = "SELECT * FROM tb_manager where name='" +
        chStr.filterStr(managerForm.getName()) + "'";           //过滤字符串中的危险字符
    ResultSet rs = conn.executeQuery(sql);
    try {                //此处需要捕获异常, 当程序出错时, 也需要将标志变量设置为0
        if (rs.next()) {
            String pwd = chStr.filterStr(managerForm.getPwd()); //获取输入的密码并过滤掉危险字符
            if (pwd.equals(rs.getString(3))) {                  //判断密码是否正确
                flag = 1;
            } else {
                flag = 0;
            }
        } else {
            flag = 0;
        }
    } catch (SQLException ex) {
        flag = 0;
    } finally {
        conn.close();                                           //关闭数据库连接
    }
    return flag;
}

```

**技巧:** 在验证用户身份时, 先判断用户名, 再判断密码, 可以防止用户输入恒等式后直接登录系统。

### 4. 防止非法用户登录系统

从网站安全的角度考虑, 仅仅上面介绍的系统登录页面并不能有效的保存系统的安全, 一旦系统主界面的地址被他人获得, 就可以通过在地址栏中输入系统的主界面地址而直接进入系统中。由于系统的 Banner 信息栏 banner.jsp 几乎包含于整个系统的每个页面, 因此这里将验证用户是否将登录的代码放置在该页中。验证用户是否登录的具体代码如下:

```


<%
String manager=(String)session.getAttribute("manager");
if (manager==null || "".equals(manager)){                //验证用户是否登录
    response.sendRedirect("login.jsp");                    //重定向网页到login.jsp页
}

```

%>

这样，当系统调用每个页面时，都会判断 session 变量 manager 是否存在，如果不存在，将页面重定向到系统登录页面。

## B.8.4 查看管理员的实现过程

 查看管理员使用的数据表：tb\_manager和tb\_purview。

管理员登录后，选择“系统设置/管理员设置”命令，进入到查看管理员列表的页面，在该页面中，将列出系统中以表格的形式显示的全部管理员及其权限信息，并提供添加管理员信息、删除管理员信息和设置管理员权限的超链接。查看管理员列表页面的运行结果，如图 B.17 所示。



图 B.17 查看管理员列表页面的运行结果

在实现系统导航菜单时，引用了 JavaScript 文件 menu.JS，该文件中包含全部实现半透明背景菜单的 JavaScript 代码。打开该 JS 文件，可以找到“管理员设置”菜单项的超链接代码，具体代码如下：

```
<a href=manager?action=managerQuery>管理员设置</a>
```

**技巧：**将页面中所涉及的 JavaScript 代码保存在一个单独的 JS 文件中，然后通过<script></script>将其引用到需要的页面，可以规范页面代码。在系统导航页面引用 menu.JS 文件的代码如下：

```
<script src="JS/menu.JS"></script>
```

从上面的 URL 地址中可以知道，查看管理员列表模块涉及到的 action 的参数值为 managerQuery，当 action= managerQuery 时，会调用查看管理员列表的方法 managerQuery()，具体代码如下：

```
if ("managerQuery".equals(action)) {
    managerQuery(request, response);           // 查询管理员及权限信息
}
```

在查看管理员列表的方法 managerQuery()中，首先调用 ManagerDAO 类中的 query()方法查询全部管理员信息，再将返回的查询结果保存到 HttpServletRequest 的对象 managerQuery 中。查看管理员列表的方法 managerQuery()的具体代码如下：

```
private void managerQuery(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String str = null;
    request.setAttribute("managerQuery", managerDAO.query(str)); //将查询结果保存到managerQuery参数中
    request.getRequestDispatcher("manager.jsp").forward(request, response); //转到显示管理员列表的页面
}
```

从 managerQuery()方法中可以看出查看管理员列表使用的 ManagerDAO 类的方法是 query()。在

query()方法中, 首先使用左连接从数据表 tb\_manager 和 tb\_purview 中查询出符合条件的数据, 然后将查询结果保存到 Collection 集合类中并返回该集合类的实例。query()方法的具体代码如下:

```

public Collection query(String queryif) {
    ManagerForm managerForm = null;           //声明ManagerForm类的对象
    ❶    Collection managercoll = new ArrayList();
    String sql = "";
    if (queryif == null || queryif == "" || queryif == "all") {    //当参数queryif的值为null、all或空时查询全部数据
    ❷    sql = "select m.*,p.sysset,p.readerset,p.bookset,p.borrowback,p.sysquery from tb_manager m left
        join tb_purview p on m.id=p.id";
        } else {
            sql="select m.*,p.sysset,p.readerset,p.bookset,p.borrowback,p.sysquery from tb_manager m left join
            tb_purview p on m.id=p.id where m.name='"+queryif+"'";    //此处需要应用左连接
        }
    ResultSet rs = conn.executeQuery(sql);    //执行SQL语句
    try {    //捕捉异常信息
        while (rs.next()) {
            managerForm = new ManagerForm();    //实例化ManagerForm类
            managerForm.setId(Integer.valueOf(rs.getString(1)));
            managerForm.setName(rs.getString(2));
            managerForm.setPwd(rs.getString(3));
            managerForm.setSysset(rs.getInt(4));
            managerForm.setReaderset(rs.getInt(5));
            managerForm.setBookset(rs.getInt(6));
            managerForm.setBorrowback(rs.getInt(7));
            managerForm.setSysquery(rs.getInt(8));
            managercoll.add(managerForm);    //将查询结果保存到Collection集合中
        }
    } catch (SQLException e) {}
    return managercoll;    //返回查询结果
}

```

### 关键代码解析

❶ Collection 接口是一个数据集合接口, 它位于与数据结构有关的 API 的最上部。通过其子接口实现 Collection 集合。

❷ 该语句应用了 MySQL 提供的左连接。在 MySQL 中左连接的语法格式如下:

```
SELECT table1.*,table2.* FROM table1 LEFT JOIN table2 ON table1.fieldname1 =table2.fieldname1;
```

接下来的工作是将 servlet 控制类中 managerQuery()方法返回的查询结果显示在查看管理员列表页 manager.jsp 中。在 manager.jsp 中首先通过 request.getAttribute()方法获取查询结果并将其保存在 Connection 集合中, 再通过循环将管理员信息以列表形式显示在页面中, 关键代码如下:

```

❶    <%@ page import="java.util.*"%>
    <%
        String flag="mr";
        Collection coll=(Collection)request.getAttribute("managerQuery");
    %>
    ❷    <% if(coll==null || coll.isEmpty()){%>
        暂无管理员信息!
    <%} else {
        //通过迭代方式显示数据
    ❸    Iterator it=coll.iterator();
        int ID=0;    //定义保存ID的变量

```



```

String name=""; //定义保存管理员名称的变量
int sysset=0; //定义保存系统设置权限的变量
int readerset=0; //定义保存读者管理权限的变量
int bookset=0; //定义保存图书管理权限的变量
int borrowback=0; //定义保存图书借还权限的变量
int sysquery=0; %>
<table width="91%" border="1" cellpadding="0" cellspacing="0" bordercolor="#FFFFFF"
bordercolordark="#D2E3E6" bordercolorlight="#FFFFFF">
<tr align="center" bgcolor="#e3F4F7">
<td width="26%">管理员名称</td>
<td width="12%">系统设置</td>
<td width="12%">读者管理</td>
<td width="12%">图书管理</td>
<td width="11%">图书借还</td>
<td width="11%">系统查询</td>
<td width="8%">权限设置</td>
<td width="8%">删除</td>
</tr>
④ <%while(it.hasNext()){
⑤ ManagerForm managerForm=(ManagerForm)it.next();
ID=managerForm.getId().intValue();
name=managerForm.getName(); //获取管理员名称
sysset=managerForm.getSysset(); //获取系统设置权限
readerset=managerForm.getReaderset(); //获取读者管理权限
bookset=managerForm.getBookset(); //获取图书管理权限
borrowback=managerForm.getBorrowback(); //获取图书借还权限
sysquery=managerForm.getSysquery(); //获取系统查询权限
%>
<tr>
<td style="padding:5px;"><%=name%></td>
<!-- 通过复选框显示管理员的权限信息，复选框没有被选中，表示该管理员不具有管理该项内容的权限-->
<td align="center"><input name="checkbox" type="checkbox" class="noborder" value="checkbox" disabled="disabled"
<%if(sysset==1){out.println("checked");}%>></td>
<td align="center"><input name="checkbox" type="checkbox" class="noborder" value="checkbox" disabled="disabled"
<%if(readerset==1){out.println("checked");}%>></td>
<td align="center"><input name="checkbox" type="checkbox" class="noborder" value="checkbox" disabled <%if
(bookset==1){out.println("checked");}%>></td>
<td align="center"><input name="checkbox" type="checkbox" class="noborder" value="checkbox" disabled <%if
(borrowback==1){out.println("checked");}%>></td>
<td align="center"><input name="checkbox" type="checkbox" class="noborder" value="checkbox" disabled <%if
(sysquery==1){out.println("checked");}%>></td>
<!-- ----->
<td align="center"> <%if(!name.equals(flag)){ %><a href="#" onClick="window.open('manager?action=
managerModifyQuery&id=<%=ID%>',", 'width=292,height=175')">权限设置</a><%} else {%>&nbsp;<%} %> </td>
<td align="center"> <%if(!name.equals(flag)){ %><a href="manager?action=managerDel&id=<%=ID%>">删除
</a><%} else {%>&nbsp;<%} %> </td>
</tr>
<% }
}%>

```

&lt;/table&gt;

### 关键代码解析

❶ `<%@ page import="packageName.className"%>`

page 指令的 import 属性用来说明在后面代码中将要使用的类和接口，这些类可以是 Sun JDK 中的类，也可以是用户自定义的类。

在 Java 里如果要载入多个包，需使用 import 分别指明，在 JSP 中也是如此。可以用一个 page 指令指定多个包（它们之间需用逗号“,”隔开），也可用多条 import 属性分别指定。

❷ import 属性是唯一一个可以在同一个页面中重复定义的 page 指令的属性。

isEmpty()方法：返回一个 boolean 对象，如果集合内未含任何元素，则返回 true。

❸ iterator()方法：返回一个 Iterator 对象，使用该方法可以用来遍历容器。

❹ hasNext()方法：检查序列中是否还有其他元素。

❺ next()方法：取得序列中的下一个元素。

## B.8.5 添加管理员的实现过程

添加管理员使用的数据表：tb\_manager。

管理员登录后，选择“系统设置/管理员设置”命令，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 B.18 所示。

图 B.18 添加管理员页面的运行结果

### 1. 设计添加管理员信息页面

添加管理员页面主要用于收集输入的管理人员信息及通过自定义的 JavaScript 函数验证输入信息是否合法，该页面中所涉及到的表单元素如表 B.12 所示。

表 B.12 添加管理员页面所涉及的表单元素

名 称	元 素 类 型	重 要 属 性	含 义
form1	form	method="post" action="manager?action=managerAdd"	表单
name	text		管理员名称
pwd	password		管理员密码
pwd1	password		确认密码
Button	button	value="保存" onClick="check(form1)"	“保存”按钮
Submit2	button	value="关闭" onClick="window.close();"	“关闭”按钮

编写自定义的 JavaScript 函数，用于判断管理员名称、管理员密码、确认密码文本框是否为空，以及两次输入的密码是否一致。程序代码如下：

```
<script language="javascript">
function check(form){
    if(form.name.value==""){
        //判断管理员名称是否为空
        alert("请输入管理员名称!");form.name.focus();return;
    }
    if(form.pwd.value==""){
        //判断管理员密码是否为空
```

```

        alert("请输入管理员密码!");form.pwd.focus();return;
    }
    if(form.pwd1.value==""){
        //判断是否输入确认密码
        alert("请确认管理员密码!");form.pwd1.focus();return;
    }
    if(form.pwd.value!=form.pwd1.value){
        //判断两次输入的密码是否一致
        alert("您两次输入的管理员密码不一致, 请重新输入!");form.pwd.focus();return;
    }
    form.submit();
    //提交表单
}
</script>

```

## 2. 修改管理员的 servlet 控制类

在添加管理员页面中, 输入合法的管理员名称及密码后, 单击“保存”按钮, 网页会访问一个 URL, 这个 URL 是 `manager?action=managerAdd`。从该 URL 地址中可以知道添加管理员信息页面涉及到的 action 的参数值为 `managerAdd`, 也就是当 `action=managerAdd` 时, 会调用添加管理员信息的方法 `managerAdd()`, 具体代码如下:

```

if ("managerAdd".equals(action)) {
    managerAdd(request, response);    // 添加管理员信息
}

```

在添加管理员信息的方法 `managerAdd()` 中, 首先需要将接收到的表单信息保存到管理员实体类 `ManagerForm` 中, 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, 则表示信息添加成功, 将页面重定向到添加信息成功的页面; 如果返回值为 2, 则表示该管理员信息已经添加, 将错误提示信息“该管理员信息已经存在!”保存到 `HttpServletRequest` 对象的 `error` 参数中, 然后将页面重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 `HttpServletRequest` 的对象 `error` 中, 并将页面重定向到错误提示页。添加管理员信息的方法 `managerAdd()` 的具体代码如下:

```

private void managerAdd(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    ManagerForm managerForm = new ManagerForm();
    managerForm.setName(request.getParameter("name"));
    managerForm.setPwd(request.getParameter("pwd"));
    int ret = managerDAO.insert(managerForm);
    if (ret == 1) {
        request.getRequestDispatcher("manager ok.jsp?para=1").forward(
            request, response);
    } else if (ret == 2) {
        request.setAttribute("error", "该管理员信息已经添加!");
        request.getRequestDispatcher("error.jsp").forward(request, response);
    } else {
        request.setAttribute("error", "添加管理员信息失败!");
        request.getRequestDispatcher("error.jsp")
            .forward(request, response);
    }
}

```

## 3. 编写添加管理员信息的 ManagerDAO 类的方法

从 `managerAdd()` 方法中可以知道添加管理员信息使用的 `ManagerDAO` 类的方法是 `insert()`。在 `insert()`

方法中首先从数据表 tb\_manager 中查询输入的管理员名称是否存在, 如果存在, 将标志变量设置为 2, 否则将输入的信息保存到管理员信息表中, 并将返回值赋给标志变量, 最后返回标志变量。insert() 方法的具体代码如下:

```
public int insert(ManagerForm managerForm) {
    String sql1="SELECT * FROM tb_manager WHERE name='"+managerForm.getName()+"'";
    ResultSet rs = conn.executeQuery(sql1); //执行SQL查询语句
    String sql = "";
    int falg = 0;
    try {
        ❶ if(rs.next()) {
            falg=2;
            //表示该管理员信息已经存在
        } else {
            sql = "INSERT INTO tb_manager (name,pwd) values('"+
                managerForm.getName() + "','" +managerForm.getPwd() +"')";
            ❷ falg = conn.executeUpdate(sql);
        }
    } catch (SQLException ex) {
        falg=0;
        //表示管理员信息添加失败
    } finally{
        conn.close();
        //关闭数据库连接
    }
    return falg;
}
```

#### 关键代码解析

❶ next()方法: 该方法为 ResultSet 接口中提供的方法, 用于移动指针到下一行。指针最初位于第一行之前, 第一次调用该方法将移动到第一行。如果存在下一行则返回 true, 否则返回 false。

❷ executeUpdate()方法是在公共模块中编写的 ConnDB 类中的方法, 该方法的返回值为 0 时, 表示数据库更新操作失败。

#### 4. 制作添加信息成功页面

这里将添加管理员信息、设置管理员权限和管理员信息删除 3 个模块操作成功的页面用一个 JSP 文件实现, 只是通过传递的参数 para 的值进行区分, 关键代码如下:


```
<%int para=Integer.parseInt(request.getParameter("para"));
switch(para){
    case 1:
        //添加信息成功时执行该代码段
        %>
        <script language="javascript">
            alert("管理员信息添加成功!");
            opener.location.reload();
            //刷新打开该窗口的页面
            window.close();
            //关闭当前窗口
        </script>
        <% break;
        //跳出switch语句
        case 2:
            //设置管理员权限成功时执行该代码段
            %>
            <script language="javascript">
                alert("管理员权限设置成功!");
                opener.location.reload();
                //刷新父窗口
                window.close();
                //关闭当前窗口
```

```

        </script>
        <% break;
        case 3:                                //删除管理员成功时执行该代码段
        %>
        <script language="javascript">
        alert("管理员信息删除成功!");
        window.location.href="manager?action=managerQuery";
        </script>
        <% break;
    }%>

```

## B.8.6 设置管理员权限的实现过程

 设置管理员权限使用的数据表：tb\_manager和tb\_purview。

管理员登录后，选择“系统设置/管理员设置”命令，进入到查看管理员列表页面，在该页面中，单击指定管理员后面的“权限设置”超链接，即可进入到权限设置页面，设置该管理员的权限。权限设置页面的运行结果如图 B.19 所示。

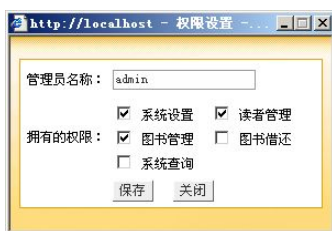


图 B.19 权限设置页面的运行结果

### 1. 在管理员列表中添加权限设置页面的入口

在“查看管理员列表”页面的管理员列表中，添加“权限设置”列，并在该列中添加以下用于打开“权限设置”页面的超链接代码。

```

<a href="#" onClick="window.open('manager?action=managerModifyQuery&id=<%=ID%>','width=292,height=175')
">权限设置</a>

```

从上面的 URL 地址中可以知道，设置管理员权限页面所涉及到的 action 的参数值为 managerModifyQuery，当 action= managerModifyQuery 时，会调用查询指定管理员权限信息的方法 managerModifyQuery()，具体代码如下：

```

if ("managerModifyQuery".equals(action)) {
    managerModifyQuery(request, response);    // 设置管理员权限时查询管理员信息
}

```

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保存到管理员实体类 ManagerForm 中；再调用 ManagerDAO 类中的 query\_update()方法，查询出指定管理员权限信息；再将返回的查询结果保存到 HttpServletRequest 的对象 managerQueryif 中。查询指定管理员权限信息的方法 managerModifyQuery()的具体代码如下：

```

private void managerModifyQuery(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    ManagerForm managerForm = new ManagerForm();
    managerForm.setIdx(Integer.valueOf(request.getParameter("id")));    // 获取并设置管理ID号
    request.setAttribute("managerQueryif", managerDAO.query_update(managerForm));
}

```

```
request.getRequestDispatcher("manager_Modify.jsp").forward(request, response); // 转到权限设置成功页面
}
```

从 `managerModifyQuery()` 中可以知道，查询指定管理员权限信息使用的 `ManagerDAO` 类的方法是 `query_update()`。在 `query_update()` 方法中，首先使用左连接从数据表 `tb_manager` 和 `tb_purview` 中查询出符合条件的数据，然后将查询结果保存到 `Collection` 集合类中，并返回该集合类。`query_update()` 方法的具体代码如下：

```
public ManagerForm query_update(ManagerForm managerForm) {
    ManagerForm managerForm1 = null;
    String sql = "select m.*,p.sysset,p.readerset,p.bookset,p.borrowback,p.sysquery from tb_manager m left join tb_purview p on m.id=p.id where m.id="+managerForm.getId() + "";
    ResultSet rs = conn.executeQuery(sql);           //执行查询语句
    try {                                           //捕捉异常信息
        while (rs.next()) {
            managerForm1 = new ManagerForm();
            managerForm1.setId(Integer.valueOf(rs.getString(1)));
            ...                                     //此处省略了设置其他属性的代码
            managerForm1.setSysquery(rs.getInt(8));
        }
    } catch (SQLException ex) {
        ex.printStackTrace();                     //输出异常信息
    } finally {
        conn.close();                             //关闭数据库连接
    }
    return managerForm1;
}
```

## 2. 设计权限设置页面

将 `Servlet` 控制类中 `managerModifyQuery()` 方法返回的查询结果，显示在设置管理员权限页 `manager_Modify.jsp` 中。在 `manager_Modify.jsp` 中，通过 `request.getAttribute()` 方法获取查询结果，并将其显示在相应的表单元素中。权限设置页面中所涉及到的表单元素如表 B.13 所示。

表 B.13 权限设置页面所涉及的表单元素

名 称	元 素 类 型	重 要 属 性	含 义
form1	form	method="post" action="manager?action=managerModify"	表单
id	hidden	value="<%=ID%>"	管理员编号
name	text	readonly="yes" value="<%=name%>"	管理员名称
sysset	checkbox	value="1" <%if(sysset==1){out.println("checked");}%>	系统设置
readerset	checkbox	value="1" <%if(readerset==1){out.println("checked");}%>	读者管理
bookset	checkbox	value="1" <%if(bookset==1){out.println("checked");}%>	图书管理
borrowback	checkbox	value="1" <%if(borrowback==1){out.println("checked");}%>	图书借还
sysquery	checkbox	value="1" <%if(sysquery==1){out.println("checked");}%>	系统查询
Button	submit	value="保存"	“保存”按钮
Submit2	button	value="关闭" onClick="window.close();"	“关闭”按钮

## 3. 修改管理员的 Servlet 控制类

在权限设置页面中设置管理员权限后，单击“保存”按钮，网页会访问一个 URL，这个 URL 是 `manager?action=managerModify`。从该 URL 地址中可以知道保存设置管理员权限信息涉及到的 `action` 的参数值为 `managerModify`，也就是当 `action=managerModify` 时，会调用保存设置管理员权限信息的

方法 `managerModify()`，具体代码如下：

```
if ("managerModify".equals(action)) {
    managerModify(request, response);    // 设置管理员权限
}
```

在保存设置管理员权限信息的方法 `managerModify()` 中，首先需要将接收到的表单信息保存到管理员实体类 `ManagerForm` 中，然后调用 `ManagerDAO` 类中的 `update()` 方法，将设置的管理员权限信息保存到权限表 `tb_purview` 中，并将返回值保存到变量 `ret` 中，如果返回值为 1，表示信息设置成功，将页面重定向到设置信息成功页面；否则，将错误提示信息“修改管理员信息失败！”保存到 `HttpServletRequest` 对象的 `error` 参数中，然后将页面重定向到错误提示信息页面。保存设置管理员权限信息的方法 `managerModify()` 的具体代码如下：

```
private void managerModify(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    ManagerForm managerForm = new ManagerForm();
    managerForm.setIdx(Integer.parseInt(request.getParameter("id")));    // 获取并设置管理员ID号
    managerForm.setName(request.getParameter("name"));    // 获取并设置管理员名称
    managerForm.setPwd(request.getParameter("pwd"));    // 获取并设置管理员密码
    managerForm.setSysset(request.getParameter("sysset") == null ? 0
        : Integer.parseInt(request.getParameter("sysset")));    // 获取并设置系统设置权限
    managerForm.setReaderset(request.getParameter("readerset") == null ? 0
        : Integer.parseInt(request.getParameter("readerset")));    // 获取并设置读者管理权限
    managerForm.setBookset(request.getParameter("bookset") == null ? 0
        : Integer.parseInt(request.getParameter("bookset")));    // 获取并设置图书管理权限
    managerForm
        .setBorrowback(request.getParameter("borrowback") == null ? 0
            : Integer.parseInt(request.getParameter("borrowback")));    // 获取并设置图书借还权限
    managerForm.setSysquery(request.getParameter("sysquery") == null ? 0
        : Integer.parseInt(request.getParameter("sysquery")));    // 获取并设置系统查询权限
    int ret = managerDAO.update(managerForm);    // 调用设置管理员权限的方法
    if (ret == 0) {
        request.setAttribute("error", "设置管理员权限失败！");    // 保存错误提示信息到error参数中
        request.getRequestDispatcher("error.jsp").forward(request, response);    // 转到错误提示页面
    } else {
        // 转到权限设置成功页面
        request.getRequestDispatcher("manager_ok.jsp?para=2").forward(request, response);
    }
}
```

#### 4. 编写保存设置管理员权限信息的 `ManagerDAO` 类的方法

从 `managerModify()` 方法中可以知道设置管理员权限时使用的 `ManagerDAO` 类的方法是 `update()`。在 `update()` 方法中，首先从数据表 `tb_manager` 中查询要设置权限的管理员是否已经存在权限信息，如果是，则修改该管理员的权限信息；如果不是，则在管理员信息表中添加该管理员的权限信息，并将返回值赋给标志变量，然后返回标志变量。`update()` 方法的具体代码如下：


```
public int update(ManagerForm managerForm) {
    String sql1 = "SELECT * FROM tb_purview WHERE id=" + managerForm.getId() + "";
    ResultSet rs = conn.executeQuery(sql1);    // 查询要设置权限的管理员的权限信息
    String sql = "";
    int falg = 0;    // 定义标志变量
    try {    // 捕捉异常信息
```

```

        if (rs.next()) { //当已经设置权限时，执行更新语句
            sql = "Update tb_purview set sysset=" + managerForm.getSysset() + ",readerset=" + managerForm.getReaderset() + ",bookset=" + managerForm.getBookset() + ",borrowback=" + managerForm.getBorrowback() + ",sysquery=" + managerForm.getSysquery() + " where id=" + managerForm.getId() + "";
        } else { //未设置权限时，执行插入语句
            sql = "INSERT INTO tb_purview values(" + managerForm.getId() + "," + managerForm.getSysset() + "," + managerForm.getReaderset() + "," + managerForm.getBookset() + "," + managerForm.getBorrowback() + "," + managerForm.getSysquery() + ")";
        }
        falg = conn.executeUpdate(sql);
    } catch (SQLException ex) {
        falg = 0; //表示设置管理员权限失败
    } finally {
        conn.close(); //关闭数据库连接
    }
    return falg;
}

```

### B.8.7 删除管理员的实现过程

 删除管理员使用的数据表：tb\_manager和tb\_purview。

管理员登录后，选择“系统设置/管理员设置”命令，进入到查看管理员列表页面，在该页面中，单击指定管理员信息后面的“删除”超链接，该管理员及其权限信息将被删除。

在查看管理员列表页面中，添加以下用于删除管理员信息的超链接代码：

```
<a href="manager?action=managerDel&id=<%=ID%>">删除</a>
```

从上面的 URL 地址中，可以知道删除管理员页所涉及到的 action 的参数值为 managerDel，当 action=managerDel 时，会调用删除管理员的方法 managerDel()，具体代码如下：

```

if ("managerDel".equals(action)) {
    managerDel(request, response); // 删除管理员
}

```

在删除管理员的方法 managerDel()中，首先需要实例化 ManagerForm 类，并用获得的 id 参数的值重新设置该类的 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。删除管理员的方法 managerDel()的具体代码如下：

```

private void managerDel(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    ManagerForm managerForm = new ManagerForm();
    managerForm.setId(Integer.valueOf(request.getParameter("id"))); // 获取并设置管理员ID号
    int ret = managerDAO.delete(managerForm); // 调用删除信息的方法delete()
    if (ret == 0) {
        request.setAttribute("error", "删除管理员信息失败！"); // 保存错误提示信息到error参数中
        request.getRequestDispatcher("error.jsp")
            .forward(request, response); // 转到错误提示页面
    } else {
        request.getRequestDispatcher("manager_ok.jsp?para=3").forward(
            request, response); // 转到删除管理员信息成功页面
    }
}

```

从 managerDel()方法中可以知道删除管理员使用的 ManagerDAO 类的方法是 delete()。在 delete()



方法中，首先将管理员信息表 tb\_manager 中符合条件的数据删除，再将权限表 tb\_purview 中的符合条件的数据删除，最后返回执行结果。delete()方法的具体代码如下：

```
public int delete(ManagerForm managerForm) {
    int flag=0;
    try{                                     //捕捉异常信息
        String sql = "DELETE FROM tb_manager where id=" + managerForm.getId() +"";
        flag = conn.executeUpdate(sql);      //执行删除管理员信息的语句
    } if (flag !=0){
        String sql1 = "DELETE FROM tb_purview where id=" + managerForm.getId() +"";
        conn.executeUpdate(sql1);          //执行删除权限信息的语句
    } catch (Exception e){
        System.out.println("删除管理员信息时产生的错误: "+e.getMessage()); //输出错误信息
    } finally{
        conn.close();                      //关闭数据库连接
    }
    return flag;
}
```

### B.8.8 单元测试

在开发完管理员模块后，为了保证程序正常运行，一定要对模块进行单元测试。单元测试在程序开发中非常重要，只有通过单元测试才能发现模块中的不足之处，才能及时的弥补程序中出现的错误。下面将对管理员模块中容易出现的错误进行分析。

在管理员模块中，最关键的环节就是验证管理员身份。下面先看一下原始的验证管理员身份的代码。

```
public int checkManager(ManagerForm managerForm) {
    int flag = 0;                          //定义标志变量
    String sql="SELECT * FROM tb_manager WHERE name='"+managerForm.getName()+
        "' and pwd='"+managerForm.getPwd()+""";
    ResultSet rs = conn.executeQuery(sql);  //执行SQL语句
    try {
        if (rs.next()) {
            flag = 1;
        } else {
            flag = 0;
        }
    } catch (SQLException ex) {
        flag = 0;
    } finally{
        conn.close();                      //关闭数据库连接
    }
    return flag;
}
```

在上面的代码中，验证管理员身份的字符串如下：

```
"SELECT * FROM tb_manager WHERE name='"+managerForm.getName()+" and pwd='"+managerForm.getPwd()+"""
```

该字符串对应的 SQL 语句为：

```
SELECT * FROM tb_manager WHERE name='管理员名称' and pwd='密码'
```

从逻辑上讲，这样的 SQL 语句并没有错误，以管理员名称和密码为条件，从数据库中查找相应的记录，如果能查询到，则认为是合法管理员。但是，这样做存在一个安全隐患，当用户在管理员名称和密码文本框中输入一个 OR 运算符及恒等式后，即使不输入正确的管理员名称和密码也可以登录到系统。例如，如果用户在管理员名称和密码文本框中分别输入 aa ' OR 'a'='a 后，上面的语句将转换为如下 SQL 语句。

```
SELECT * FROM tb_user WHERE name=' aa ' OR 'a'='a ' AND pwd=' aa ' OR 'a'='a '
```

由于表达式'a'='a'的值为真，系统将查出全部管理员信息，所以即使用户输入错误的管理员名称和密码也可以轻松登录系统。因此，这里采用了先过滤掉输入字符串中的危险字符，再分别判断输入的管理员名称和密码是否正确的方法。修改后的验证管理员身份的代码如下：

```
public int checkManager(ManagerForm managerForm) {
    int flag = 0;
    ChStr chStr=new ChStr(); //实例化ChStr类的一个对象
    String sql = "SELECT * FROM tb_manager where name='" +
        chStr.filterStr(managerForm.getName()) + "'";
    ResultSet rs = conn.executeQuery(sql); //执行SQL语句
    try {
        if (rs.next()) {
            String pwd = chStr.filterStr(managerForm.getPwd()); //获取输入的密码并过滤掉危险字符
            if (pwd.equals(rs.getString(3))) { //判断密码是否正确
                flag = 1;
            } else {
                flag = 0;
            }
        } else {
            flag = 0;
        }
    } catch (SQLException ex) {
        flag = 0;
    } finally {
        conn.close(); //关闭数据库连接
    }
    return flag;
}
```

## B.9 图书借还模块设计

### B.9.1 图书借还模块概述

图书借还模块主要包括图书借阅、图书续借、图书归还、图书借阅查询、借阅到期提醒和图书借阅排行 6 个功能。在图书借阅模块中的用户，只有一种身份，那就是操作员，通过该身份可以进行图书借还等相关操作。图书借还模块的用例图如图 B.20 所示。

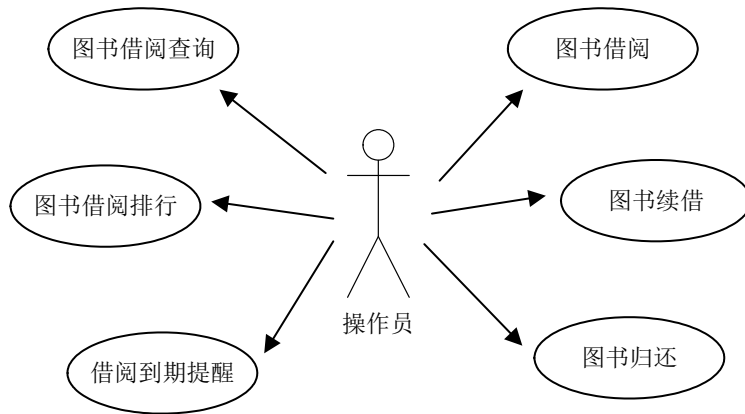


图 B.20 图书借还模块的用例图

## B.9.2 图书借还模块技术分析

在实现图书借还模块时，需要编写图书借还模块对应的 ActionForm 类和 Servlet 控制类。下面将详细介绍如何编写图书借还模块的 ActionForm 类和 Servlet 控制类。

### 1. 编写图书借还的实体类

在图书借还模块中涉及到的数据表是 tb\_borrow（图书借阅信息表）、tb\_bookinfo（图书信息表）和 tb\_reader（读者信息表），这 3 个数据表间通过相应的字段进行关联，如图 B.21 所示。

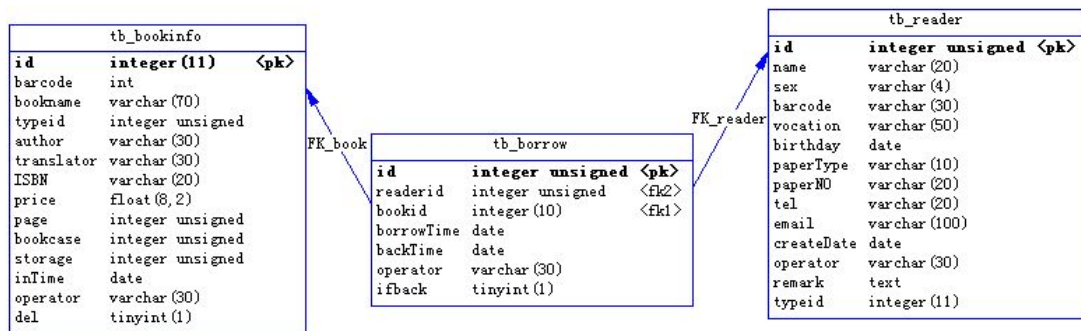


图 B.21 图书借还管理模块各表间关系图

通过以上 3 个表可以获得图书借还信息，根据这些信息来创建图书借还模块的实体类，名称为 BorrowForm，具体实现方法请读者参见 B.8.2 节“管理员模块技术分析”。

### 2. 编写图书借还的 Servlet 控制类

图书借还模块的 Servlet 控制类 Borrow 继承了 HttpServlet 类，在该类中，首先需要在构造方法中实例化图书借还管理模块的 BookDAO 类、BorrowDAO 类和 ReaderDAO 类（这些类用于实现与数据库的交互），然后编写 doGet()和 doPost()方法，在这两个方法中根据 request 的 getParameter()方法获取的 action 参数值执行相应方法，由于这两个方法中的代码相同，所以只需在第一个方法 doGet()中写相应代码，在另一个方法 doPost()中调用 doGet()方法即可。

图书借还模块 Servlet 控制类的关键代码如下：

```


public class Borrow extends HttpServlet {
    /**
     * *****在构造方法中实例化Borrow类中应用的持久层类的对象*****
     */
    private BorrowDAO borrowDAO = null;
    private ReaderDAO readerDAO=null;
  
```

```

private BookDAO bookDAO=null;
private ReaderForm readerForm=new ReaderForm();
public Borrow() {
    this.borrowDAO = new BorrowDAO();
    this.readerDAO=new ReaderDAO();
    this.bookDAO=new BookDAO();
}
/*****
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    String action =request.getParameter("action");    //获取action参数的值
    if(action==null||"".equals(action)){
        request.setAttribute("error","您的操作有误! ");
        request.getRequestDispatcher("error.jsp").forward(request, response);
    }else if("bookBorrowSort".equals(action)){
        bookBorrowSort(request,response);
    }else if("bookborrow".equals(action)){
        bookborrow(request,response);    //图书借阅
    }else if("bookrenew".equals(action)){
        bookrenew(request,response);    //图书续借
    }else if("bookback".equals(action)){
        bookback(request,response);    //图书归还
    }else if("Bremind".equals(action)){
        breminde(request,response);    //借阅到期提醒
    }else if("borrowQuery".equals(action)){
        borrowQuery(request,response);    //借阅信息查询
    }
}
..... //此处省略了该类中其他方法, 这些方法将在后面的具体过程中给出
}

```

### B.9.3 图书借阅的实现过程

 图书借阅使用的数据表: tb\_borrow、tb\_bookinfo和tb\_reader。

管理员登录后, 选择“图书借还/图书借阅”命令, 进入到图书借阅页面, 在该页面中的“读者条形码”文本框中输入读者的条形码 (如: 20110224000001) 后, 单击“确定”按钮, 系统会自动检索出该读者的基本信息和未归还的借阅图书信息。如果找到对应的读者信息, 就将其显示在页面中, 此时输入图书的条形码或图书名称后, 单击“确定”按钮, 借阅指定的图书, 图书借阅页面的运行结果如图 B.22 所示。

博研图书馆管理系统  
BO YAN LIBRARY MANAGE SYSTEM

当前登录用户: mr

2011年02月25日 星期五 14:57:14 | 首页 | 系统设置 | 读者管理 | 图书管理 | 图书借还 | 系统查询 | 更改口令 | 退出系统

当前位置: 图书借还 > 图书借还 >>>

**图书借还**

读者验证 读者条形码: 20110224000001 确定

姓 名: 琦琦 性 别: 女 读者类型: 学生

证件类型: 身份证 证件号码: 220104201007100001 可借数量: 1 册

添加的依据: ☒ 图书条形码 ☐ 图书名称 确定 完成借还

图书名称	借阅时间	应还时间	出版社	书架	定价(元)
Java Web开发实战宝典	2011-02-24	2011-03-16	清华大学出版社	左A-1	89.0

Copyright © 2011 www.\*\*\*\*\*.com 长春市\*\*\*\*\*有限公司  
本站请使用IE6.0或以上版本 1024\*768为最佳显示效果

图 B.22 图书借阅页面的运行结果

### 1. 设计图书借阅页面

图书借阅页面总体上可以分为两个部分：一部分用于查询并显示读者信息；另一部分用于显示读者的借阅信息和添加读者借阅信息。图书借阅页面在 Dreamweaver 中的设计效果如图 B.23 所示。

**图书借还**

读者验证 读者条形码: <%=barcode%> 确定

姓 名: <%=name%> 性 别: <%=sex%> 读者类型: <%=typename%>

证件类型: <%=paperType%> 证件号码: <%=paperNO%> 可借数量: <%=number%> 册

添加的依据: ☒ 图书条形码 ☐ 图书名称 确定 完成借还

图书名称	借阅时间	应还时间	出版社	书架	定价(元)
<%=bookName%>	<%=borrowTime%>	<%=returnTime%>	<%=publisher%>	<%=shelf%>	<%=price%>

图 B.23 在 Dreamweaver 中图书借阅页面的设计效果

由于系统要求一个读者只能同时借阅一定数量的图书，并且该数量由读者类型表 tb\_readerType 中的可借数量 number 决定，所以这里编写了自定义的 JavaScript 函数 checkbook(), 用于判断当前选择的读者是否还可以借阅新的图书，同时该函数还具有判断是否输入图书条形码或图书名称的功能，代码如下：

```
<script language="javascript">
function checkbook(form){
    if(form.barcode.value==""){
        //判断是否输入读者条形码
        alert("请输入读者条形码!");form.barcode.focus();return;
    }
    if(form.inputkey.value==""){
        //判断查询关键字是否为空
        alert("请输入查询关键字!");form.inputkey.focus();return;
    }
    if(form.number.value-form.borrowNumber.value<=0){
        //判断是否可以再借阅其他图书
        alert("您不能再借阅其他图书了!");return;
    }
}
```

```

        form.submit();                                //提交表单
    }
</script>

```

**技巧:** 在 JavaScript 中比较两个数值型文本框的值时, 不使用运算符“==”, 而是将这两个值相减, 再判断其结果。

## 2. 修改图书借阅的 Servlet 控制类

在图书借阅页面中的“读者条形码”文本框中输入条形码后, 单击“确定”按钮, 或者在“图书条形码”/“图书名称”文本框中输入图书条形码或图书名称后, 单击“确定”按钮, 网页会访问一个 URL, 这个 URL 是 borrow?action=bookborrow。从该 URL 地址中可以知道图书借阅模块涉及到的 action 的参数值为 bookborrow, 也就是当 action=bookborrow 时, 会调用图书借阅的方法 bookborrow(), 具体代码如下:

```

if("bookborrow".equals(action)){
    bookborrow(request,response);    //图书借阅
}

```

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实体类 (ReaderForm) 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 ReaderDAO 类中的 queryM()方法查询读者信息, 并将查询结果保存在 ReaderForm 的对象 reader 中; 最后将 reader 保存到 HttpServletRequest 的对象 readerinfo 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 HttpServletRequest 的对象 borrowinfo 中。

(3) 首先获取查询条件 (是按图书条形码还是按图书名称查询) 和查询关键字, 如果查询关键字不为空时, 调用 BookDAO 类的 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 BorrowDAO 类的 insertBorrow()方法添加图书借阅信息 (如果添加图书借阅信息成功, 则将当前读者条形码保存到 HttpServletRequest 对象的 bar 参数中, 并且返回到图书借阅成功页面; 否则将错误信息“添加借阅信息失败!”保存到 HttpServletRequest 的对象的 error 参数中, 并将页面重定向到错误提示页), 否则将错误提示信息“没有该图书!”保存到 HttpServletRequest 对象的 error 参数中。

图书借阅的方法 bookborrow()的具体代码如下:

```

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

    ReaderForm readerForm=new ReaderForm();
    readerForm.setBarcode(request.getParameter("barcode"));           //获取读者条形码
    ❶ ReaderForm reader = (ReaderForm) readerDAO.queryM(readerForm);
    request.setAttribute("readerinfo", reader);                       //保存读者信息到readerinfo中
                                                                    //查询读者的借阅信息

    request.setAttribute("borrowinfo",borrowDAO.borrowinfo(request.getParameter("barcode")));
    /*****完成借阅*****/

    String f = request.getParameter("f");                             //获取查询方式
    String key = request.getParameter("inputkey");                   //获取查询关键字
    if (key != null && !key.equals("")) {                             //当图书名称或图书条形码不为空时
        String operator = request.getParameter("operator");          //获取操作员
        ❷ BookForm bookForm=bookDAO.queryB(f, key);
        if (bookForm!=null){
            int ret = borrowDAO.insertBorrow(reader, bookDAO.queryB(f, key), operator);
            if (ret == 1) {
                ❸ request.setAttribute("bar", request.getParameter("barcode"));
            }
        }
    }
}

```

```

        request.getRequestDispatcher("bookBorrow_ok.jsp").forward(request, response); //转到借阅成功页面
    } else {
        request.setAttribute("error", "添加借阅信息失败!");
        request.getRequestDispatcher("error.jsp").forward(request, response); //转到错误提示页面
    }
} else {
    request.setAttribute("error", "没有该图书!");
    request.getRequestDispatcher("error.jsp").forward(request, response); //转到错误提示页面
}
} else {
    request.getRequestDispatcher("bookBorrow.jsp").forward(request, response); //转到图书借阅页面
}
}
}

```

### 3. 编写借阅图书的 BorrowDAO 类的方法

从 bookborrow() 方法中可以知道, 保存借阅图书信息时使用的 BorrowDAO 类的方法是 insertBorrow()。在 insertBorrow() 方法中, 首先从数据表 tb\_bookinfo 中查询出借阅图书的 ID; 然后再获取系统日期(用于指定借阅时间), 并计算归还时间; 再将图书借阅信息保存到借阅信息表 tb\_borrow 中。图书借阅的方法 insertBorrow() 的代码如下:

```

public int insertBorrow(ReaderForm readerForm, BookForm bookForm, String operator) {
    /******* 获取系统日期 *****/
    Date dateU = new Date();
    java.sql.Date date = new java.sql.Date(dateU.getTime());
    /*******

    String sql1 = "select t.days from tb_bookinfo b left join tb_booktype t on b.typeid=t.id where b.id="+bookForm.getId()+"";
    ResultSet rs = conn.executeQuery(sql1); //执行查询语句
    int days = 0;
    try {
        if (rs.next()) {
            days = rs.getInt(1); //获取可借阅天数
        }
    } catch (SQLException ex) {
    }

    /******* 计算归还时间 *****/
    ❶ String date_str = String.valueOf(date);
    ❷ String dd = date_str.substring(8, 10);
    String DD = date_str.substring(0, 8) + String.valueOf(Integer.parseInt(dd) + days);
    java.sql.Date backTime = java.sql.Date.valueOf(DD);
    /*******

    String sql = "Insert into tb_borrow (readerid, bookid, borrowTime, backTime, operator) values (" + readerForm.getId() + ", "
    + bookForm.getId() + ", " + date + ", " + backTime + ", " + operator + ")";
    int falg = conn.executeUpdate(sql); //执行插入语句
    conn.close(); //关闭数据库连接
    return falg;
}

```

#### 关键代码解析

- ❶ String.valueOf(date): 用于返回 date 的字符串表现形式。
- ❷ substring() 方法: 用于获得字符串的子字符串。该方法的语法格式如下:



substring(int start)或

substring(int start,int end)

功能：返回原字符串中从 start 开始直到字符串尾或者直到 end 之间的所有字符所组成的新串。

参数说明如下。

start：表示起始位置的值，该位置从 0 开始计算。

end：表示结束位置的值，但不包括此位置。

## B.9.4 图书续借的实现过程

图书续借使用的数据表：tb\_borrow、tb\_bookinfo和tb\_reader。

管理员登录后，选择“图书借还”/“图书续借”命令，进入到图书续借页面，在该页面中的“读者条形码”文本框中输入读者的条形码（如 20110224000001）后，单击“确定”按钮，系统会自动检索出该读者的基本信息和未归还的借阅图书信息。如果找到对应的读者信息，则将其显示在页面中，此时单击“续借”超链接，即可续借指定图书（即将该图书的归还时间延长到指定日期，该日期由续借日期加上该书的可借天数计算得出）。图书续借页面的运行结果如图 B.24 所示。

博研图书馆管理系统  
BO YAN LIBRARY MANAGE SYSTEM

当前登录用户: mr

2011年02月25日 星期五 15:22:18

首页 | 系统设置 | 读者管理 | 图书管理 | 图书借还 | 系统查询 | 更改口令 | 退出系统

当前位置: 图书借还 > 图书续借 >>>

**图书续借**

读者验证 读者条形码: 20110224000001 确定

姓 名: 琦琦 性 别: 女 读者类型: 学生

证件类型: 身份证 证件号码: 220104201007100001 可借数量: 1 册

图书名称	借阅时间	应还时间	出版社	书架	定价(元)	完成续借
Java Web开发实战宝典	2011-02-24	2011-03-16	清华大学出版社	左A-1	89.0	续借

Copyright © 2011 www.\*\*\*\*\*.com 长春市\*\*\*\*\*有限公司  
本站请使用IE6.0或以上版本 1024\*768为最佳显示效果

图 B.24 图书续借页面的运行结果

### 1. 设计图书续借页面

图书续借页面的设计方法同图书借阅页面类似，所不同的是，在图书续借页面中没有添加借阅图书的功能，而是添加了“续借”超链接。图书续借页面在 Dreamweaver 中的设计效果如图 B.25 所示。

**图书续借**

读者验证 读者条形码: <%=barcode%> 确定

姓 名: <%=name%> 性 别: <%=sex%> 读者类型: <%=typename%>

证件类型: <%=paperType%> 证件号码: <%=paperNO%> 可借数量: <%=number%> 册

图书名称	借阅时间	应还时间	出版社	书架	定价(元)	完成续借
<%=bookName%>	<%=borrowTime%>	<%=returnTime%>	<%=publisher%>	<%=shelf%>	<%=price%>	续借

图 B.25 在 Dreamweaver 中的图书续借页面的设计效果



在单击“续借”超链接时,还需要将读者条形码和借阅ID号一起传递到图书续借的Servlet控制类中,代码如下:

```
<a href="borrow?action=bookrenew&barcode=<%=barcode%>&id=<%=id%>">续借</a>
```

## 2. 修改图书续借的Servlet控制类

在图书续借页面中的“读者条形码”文本框中输入条形码后,单击“确定”按钮,网页会访问一个URL,这个URL是borrow?action=bookrenew。从该URL地址中可以知道图书续借模块涉及到的action的参数值为bookrenew,也就是当action=bookrenew时,会调用图书续借的方法bookrenew(),具体代码如下:

```
if("bookrenew".equals(action)){
    bookrenew(request,response); //图书续借
}
```

实现图书续借的方法bookback()需要分以下3个步骤进行:

(1) 首先需要实例化读者信息所对应的ActionForm(ReaderForm)的对象,然后将该对象的setBarcode()方法设置为从页面中获取读者条形码的值,再调用ReaderDAO类中的queryM()方法查询读者信息,并将查询结果保存在ReaderForm的对象reader中,最后将reader保存到HttpServletRequest的对象readerinfo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HttpServletRequest的对象borrowinfo中。

(3) 首先判断是否从页面中传递了借阅ID号,如果是,则获取从页面中传递的借阅ID号,然后判断该id值是否大于0,如果大于0,则调用BorrowDAO类的renew()方法执行图书续借操作。如果图书续借操作执行成功,则将当前读者条形码保存到HttpServletRequest对象的bar参数中,并且返回到图书续借成功页面,否则将错误信息“图书续借失败!”保存到HttpServletRequest对象的error参数中,并将页面重定向到错误提示页。

图书续借的方法bookrenew()的具体代码如下:

```
private void bookrenew(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    /*******根据输入的读者条形码查询读者信息******/
    readerForm.setBarcode(request.getParameter("barcode"));
    ReaderForm reader = (ReaderForm) readerDAO.queryM(readerForm);
    request.setAttribute("readerinfo", reader);
    /*******查询读者的借阅信息******/
    request.setAttribute("borrowinfo", borrowDAO.borrowinfo(request.getParameter("barcode")));
    if(request.getParameter("id")!=null){
        int id = Integer.parseInt(request.getParameter("id"));
        if(id > 0) {
            //执行续借操作
            int ret = borrowDAO.renew(id);
            //调用renew()方法完成图书续借
            if(ret == 0) {
                request.setAttribute("error", "图书续借失败!");
                request.getRequestDispatcher("error.jsp").forward(request, response); //转到错误提示页
            } else {
                request.setAttribute("bar", request.getParameter("barcode"));
                request.getRequestDispatcher("bookRenew_ok.jsp").forward(request, response); //转到借阅成功页面
            }
        }
    }
} else {
    request.getRequestDispatcher("bookRenew.jsp").forward(request, response);
}
```

```

    }
}

```

### 3. 编写续借图书的 BorrowDAO 类的方法

从 bookrenew()方法中可以知道,保存图书续借信息时使用的 BorrowDAO 类的方法是 renew()。在 renew()方法中,首先根据借阅 ID 号从数据表 tb\_borrow 中查询出当前借阅信息的读者 ID 和图书 ID,然后再获取系统日期(用于指定归还时间),再将图书归还信息保存到图书归还信息表 tb\_giveback 中,最后将图书借阅信息表中该记录的“是否归还”字段 ifback 的值设置为 1,表示已经归还。图书归还的方法 back()的代码如下:


```

public int renew(int id){
    String sql0="SELECT bookid FROM tb_borrow WHERE id="+id+"";
    ResultSet rs1=conn.executeQuery(sql0);           //执行查询语句
    int flag=0;
    try {
        if (rs1.next()) {
            /*******获取系统日期*****/
            Date dateU = new Date();
            java.sql.Date date = new java.sql.Date(dateU.getTime());
            /*******/
            String sql1 = "select t.days from tb_bookinfo b left join tb_booktype t on b.typeid=t.id where b.id=" +
                rs1.getInt(1) + """;
            ResultSet rs = conn.executeQuery(sql1);           //执行查询语句
            int days = 0;
            try {
                if (rs.next()) {
                    days = rs.getInt(1);           //获取图书的可借天数
                }
            } catch (SQLException ex) {}
            /*******计算归还时间*****/
            String date_str = String.valueOf(date);
            String dd = date_str.substring(8, 10);
            String DD = date_str.substring(0, 8) + String.valueOf(Integer.parseInt(dd) + days);
            java.sql.Date backTime = java.sql.Date.valueOf(DD);

            /*******/
            String sql = "UPDATE tb_borrow SET backtime=" + backTime + " where id=" + id + "";
            flag = conn.executeUpdate(sql);           //执行更新语句
        }
    } catch (Exception ex1) {}
    conn.close();           //关闭数据库连接
    return flag;
}

```

## B.9.5 图书归还的实现过程

 图书归还使用的数据表: tb\_borrow、tb\_bookinfo和tb\_reader。

管理员登录后,选择“图书借还”/“图书归还”命令,进入到图书归还页面,在该页面中的“读者条形

码”文本框中输入读者的条形码（如：20110224000001）后，单击“确定”按钮，系统会自动检索出该读者的基本信息和未归还的借阅图书信息。如果找到对应的读者信息，则将其显示在页面中，此时单击“归还”超链接，即可将指定图书归还。图书归还页面的运行结果如图 B.26 所示。

博研图书馆管理系统  
BO YAN LIBRARY MANAGE SYSTEM

2011年02月25日 星期五 15:32:43 首页 | 系统设置 | 读者管理 | 图书管理 | 图书借还 | 系统查询 | 更改口令 | 退出系统

当前登录用户: mr

当前位置: 图书借还 > 图书归还 >>>

**图书归还**

读者验证 读者条形码: 20110224000001 确定

姓 名: 琦琦 性 别: 女 读者类型: 学生

证件类型: 身份证 证件号码: 220104201007100001 可借数量: 1 册

图书名称	借阅时间	应还时间	出版社	书架	定价(元)	完成归还
Java Web开发实战宝典	2011-02-24	2011-03-16	清华大学出版社	左A-1	89.0	归还

Copyright © 2011 www.\*\*\*\*\*.com 长春市\*\*\*\*\*有限公司  
本站请使用IE6.0或以上版本 1024\*768为最佳显示效果

图 B.26 图书归还页面的运行结果

### 1. 设计图书归还页面

图书归还页面的设计方法同图书续借页面类似，所不同的是，将图书续借页面中的“续借”超链接转化为“归还”超链接。在单击“归还”超链接时，也需要将读者条形码、借阅 ID 号和操作员一同传递到图书归还的 Servlet 控制类中，代码如下：

```
<a href="borrow?action=bookback&barcode=<%=barcode%>&id=<%=id%>&operator=<%=manager%>">归还</a>
```

### 2. 修改图书归还的 Servlet 控制类

在图书归还页面中的“读者条形码”文本框中输入条形码后，单击“确定”按钮，网页会访问一个 URL，这个 URL 是 borrow?action=bookback。从该 URL 地址中可以知道图书归还模块涉及到的 action 的参数值为 bookback，也就是当 action= bookback 时，会调用图书归还的方法 bookback()，具体代码如下：

```
if("bookback".equals(action)){
    bookback(request,response);           //图书归还
}
```

实现图书归还的方法 bookback()与实现图书续借的方法 bookrenew ()基本相同，所不同的是如果从页面中传递的借阅 ID 号大于 0，则调用 BorrowDAO 类的 back()方法执行图书归还操作，并且需要获取页面中传递的操作员信息。图书归还的方法 bookback()的关键代码如下：


```
int id = Integer.parseInt(request.getParameter("id"));
String operator=request.getParameter("operator");           //获取页面中传递的操作员信息
if(id > 0) { //执行归还操作
    int ret = borrowDAO.back(id,operator);                   //调用back()方法执行图书归还操作
    ... //此处省略了其他代码
}
```

### 3. 编写归还图书的 BorrowDAO 类的方法

从 bookback() 方法中可以知道, 保存归还图书信息时使用的 BorrowDAO 类的方法是 back()。在 back() 方法中, 首先根据借阅 ID 号从数据表 tb\_borrow 中查询出当前借阅信息的读者 ID 和图书 ID; 然后再获取系统日期 (用于指定归还时间), 再将图书归还信息保存到图书归还信息表 tb\_giveback 中; 最后将图书借阅信息表中该记录的“是否归还”字段 ifback 的值设置为 1, 表示已经归还。图书归还的方法 back() 的代码如下:

```
public int back(int id,String operator){
    String sql0="SELECT readerid,bookid FROM tb_borrow WHERE id="+id+"";
    ResultSet rs1=conn.executeQuery(sql0);           //执行查询语句
    int flag=0;
    try {
        if (rs1.next()) {
            /*****获取系统日期*****/
            Date dateU = new Date();
            java.sql.Date date = new java.sql.Date(dateU.getTime());
            /*****/
            int readerid=rs1.getInt(1);
            int bookid=rs1.getInt(2);
            String sql1="INSERT INTO tb_giveback (readerid,bookid,backTime,operator) VALUES("+
                readerid+", "+bookid+", "+date+", "+operator+"";
            int ret=conn.executeUpdate(sql1);           //执行插入操作
            if(ret==1){
                String sql2 = "UPDATE tb_borrow SET ifback=1 where id=" + id +"";
                flag = conn.executeUpdate(sql2);       //执行更新操作
            }else{
                flag=0;
            }
        }
    } catch (Exception ex1) {}
    conn.close();           //关闭数据库连接
    return flag;
}
```

## B.9.6 图书借阅查询的实现过程

 图书借阅查询使用的数据表: tb\_borrow、tb\_bookinfo 和 tb\_reader。

管理员登录后, 选择“系统查询”/“图书借阅查询”命令, 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, 同时还可以按指定字段及时间段进行综合查询。图书借阅查询页面的运行结果如图 B.22 所示。



图 B.22 图书借阅查询页面的运行结果

1. 设计图书借阅查询页面

图书借阅查询页面主要用于收集查询条件和显示查询结果，并通过自定义的 JavaScript 函数验证输入的查询条件是否合法，该页面中所涉及到的表单元素如表 B.14 所示。

表 B.14 图书借阅查询页面所涉及的表单元素

名 称	元 素 类 型	重 要 属 性	含 义
myform	form	method="post" action="borrow?action=borrowQuery"	表单
flag	checkbox	value="a" checked	选择查询依据
flag	checkbox	value="b"	借阅时间
f	select	<option value="barcode">图书条形码</option> <option value="bookname">图书名称</option> <option value="readerbarcode">读者条形码</option> <option value="readername">读者名称</option>	查询字段
key	text	size="50"	关键字
sdate	text		开始日期
edate	text		结束日期
Submit	submit	value="查询" onClick="return check(myform)"	“查询”按钮

编写自定义的 JavaScript 函数 check(), 用于判断是否选择了查询方式及当选择按时间段进行查询时，判断输入的日期是否合法。代码如下：

```
<script language="javascript">
function check(myform){
❶    if(myform.flag[0].checked==false && myform.flag[1].checked==false){
        alert("请选择查询方式!");return false;
    }
    if (myform.flag[1].checked){
        if(myform.sdate.value==""){
            //判断是否输入开始日期
            alert("请输入开始日期");myform.sdate.focus();return false;
        }
❷    if(CheckDate(myform.sdate.value)){
            //判断开始日期的格式是否正确
            alert("您输入的开始日期不正确（如：2011-02-14）\n 请注意闰年!");
            myform.sDate.focus();return false;
        }
    }
}
```

```

        if(myform.edate.value==""){
            //判断是否输入结束日期
            alert("请输入结束日期");myform.edate.focus();return false;
        }
        if(CheckDate(myform.edate.value)){
            //判断结束日期的格式是否正确
            alert("您输入的结束日期不正确（如：2011-02-14）\n 请注意闰年!");
            myform.edate.focus();return false;
        }
    }
}
</script>

```

### 关键代码解析

- ❶ myform.flag[0].checked: 表示复选框是否被选中, 值为 true, 表示被选中, 值为 false, 表示未被选中。
- ❷ CheckDate(): 为自定义的 JavaScript 函数, 该函数用于验证日期, 保存在 JS\function.js 文件中。

## 2. 修改图书借阅查询的 Servlet 控制类

在图书借阅查询页面中, 选择查询方式及查询关键字后, 单击“查询”按钮, 网页会访问一个 URL, 这个 URL 是 borrow?action=borrowQuery。从该 URL 地址中可以知道图书借阅查询模块涉及到的 action 的参数值为 borrowQuery, 也就是当 action=borrowQuery 时, 会调用图书借阅查询的方法 borrowQuery(), 具体代码如下:

```

if("borrowQuery".equals(action)){
    borrowQuery(request,response); //借阅信息查询
}

```

在图书借阅查询的方法 borrowQuery() 中, 首先获取表单元复选框 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 HttpServletRequest 对象的 borrowQuery 参数中。图书借阅查询的方法 bookborrow() 的具体代码如下:

```

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

    String str=null;
    ❶ String flag[]=request.getParameterValues("flag"); //获取复选框的值
    /*****以指定字段为条件时查询的字符串*****/

    if (flag!=null){
        String aa = flag[0];
        if ("a".equals(aa)) {
            if (request.getParameter("f") != null) {
                str = request.getParameter("f") + " like '%" + request.getParameter("key") + "%'";
            }
        }

        /*****/
        /****以指定时间段为条件时查询的字符串*****/

        if ("b".equals(aa)) {
            String sdate = request.getParameter("sdate"); //获取开始日期
            String edate = request.getParameter("edate"); //获取结束日期
            if (sdate != null && edate != null) {
                str = "borrowTime between '" + sdate + "' and '" + edate + "'";
            }
        }

        /*****/
    }
}

```

```

/*****将指定的字段条件、时间段条件组合后查询的字符串*****/
②    if (flag.length == 2) {
        if (request.getParameter("f") != null) {
            str = request.getParameter("f") + " like '%" + request.getParameter("key") + "%'";
        }
        String sdate = request.getParameter("sdate");    //获取开始日期
        String edate = request.getParameter("edate");    //获取结束日期
        String str1 = null;
        if (sdate != null && edate != null) {
            str1 = "borrowTime between '" + sdate + "' and '" + edate + "'";
        }
        str = str + " and borr." + str1;
    }
}

/*****/
request.setAttribute("borrowQuery", borrowDAO.borrowQuery(str));
request.getRequestDispatcher("borrowQuery.jsp").forward(request, response);    //转到查询借阅信息页面
}

```

### 3. 编写图书借阅查询的 BorrowDAO 类的方法

从 borrowQuery()方法中可以知道,图书借阅查询时使用的 BorrowDAO 类的方法是 borrowQuery()。在 borrowQuery()方法中,首先根据参数 strif 的值确定要执行的 SQL 语句,然后将查询结果保存到 Collection 集合类中,并返回该集合类的实例。图书借阅查询的方法 borrowQuery()的代码如下:

```

public Collection borrowQuery(String strif) {
    String sql = "";
    if (strif != "all" && strif != null && strif != "") { //当查询条件不为空时
        sql = "select * from (select borr.borrowTime,borr.backTime,book.barcode,book.bookname,r.name readername,
r.barcode readerbarcode,borr.ifback from tb_borrow borr join tb_bookinfo book on book.id=borr.bookid join tb_reader r on
r.id=borr.readerid) as borr where borr." + strif + "";
    } else { //当查询条件为空时
        sql = "select * from (select borr.borrowTime,borr.backTime,book.barcode,book.bookname,r.name readername,
r.barcode readerbarcode,borr.ifback from tb_borrow borr join tb_bookinfo book on book.id=borr.bookid join tb_reader r on
r.id=borr.readerid) as borr"; //查询全部数据
    }
    ResultSet rs = conn.executeQuery(sql); //执行查询语句
    Collection coll = new ArrayList(); //初始化Collection的实例
    BorrowForm form = null;
    try { //捕捉异常信息
        while (rs.next()) {
            form = new BorrowForm();
            form.setBorrowTime(rs.getString(1)); //获取并设置借阅时间属性
            ... //此处省略了获取并设置其他属性信息的代码
            coll.add(form); //将查询结果保存到Collection集合类中
        }
    } catch (SQLException ex) {
        System.out.println(ex.getMessage()); //输出异常信息
    }
    conn.close(); //关闭数据库连接
    return coll;
}

```

}

### B.9.7 单元测试

在开发完成图书借阅模块并测试时，会发现以下问题：当管理员进入到“图书借阅”页面后，在“读者条形码”文本框中输入读者条形码（如 20110224000001），并单击其后面的“确定”按钮，即可调出该读者的基本信息，这时，在“添加依据”文本框中输入相应的图书信息后，单击其后面的“确定”按钮，页面将直接返回到图书借阅首页，当再次输入读者条形码后，就可以看到刚刚添加的借阅信息。由于在图书借阅时，可能存在同时借阅多本图书的情况，这样将给操作员带来不便。

下面先看一下原始的完成借阅的代码：

```

if (key != null && !key.equals("")) { //当图书名称或图书条形码不为空时
    String operator = request.getParameter("operator"); //获取操作员
    BookForm bookForm=bookDAO.queryB(f, key);
    if (bookForm!=null){
        int ret = borrowDAO.insertBorrow(reader, bookDAO.queryB(f, key), operator);
        if (ret == 1) {
            request.getRequestDispatcher("bookBorrow_ok.jsp").forward(request, response); //转到借阅成功页面
        } else {
            request.setAttribute("error", "添加借阅信息失败!");
            request.getRequestDispatcher("error.jsp").forward(request, response); //转到错误提示页面
        }
    } else {
        request.setAttribute("error", "没有该图书!");
        request.getRequestDispatcher("error.jsp").forward(request, response); //转到错误提示页面
    }
} else {
    request.getRequestDispatcher("bookBorrow.jsp").forward(request, response); //转到图书借阅页面
}

```

从上面的代码中可以看出，在转到图书借阅页面前，并没有保存读者条形码，这样在返回图书借阅页面时，就会出现直接返回到图书借阅首页的情况。解决该问题的方法是在“request.getRequestDispatcher("bookBorrow\_ok.jsp").forward(request, response);”语句的前面添加以下语句：

```
request.setAttribute("bar", request.getParameter("barcode"));
```

将读者条形码保存到 HttpServletRequest 对象的 bar 参数中，这样，在完成一本图书的借阅后，将不会直接退出到图书借阅首页，而是可以直接进行下一次借阅操作。修改后的完成借阅的代码如下：

```

if (key != null && !key.equals("")) { //当图书名称或图书条形码不为空时
    String operator = request.getParameter("operator"); //获取操作员
    BookForm bookForm=bookDAO.queryB(f, key);
    if (bookForm!=null){
        int ret = borrowDAO.insertBorrow(reader, bookDAO.queryB(f, key), operator);
        if (ret == 1) {
            request.setAttribute("bar", request.getParameter("barcode"));
            request.getRequestDispatcher("bookBorrow_ok.jsp").forward(request, response); //转到借阅成功页面
        } else {
            request.setAttribute("error", "添加借阅信息失败!");
            request.getRequestDispatcher("error.jsp").forward(request, response); //转到错误提示页面
        }
    }
}

```



```

    }else{
        request.setAttribute("error", "没有该图书!");
        request.getRequestDispatcher("error.jsp").forward(request, response);    //转到错误提示页面
    }
    }else{
        request.getRequestDispatcher("bookBorrow.jsp").forward(request, response);    //转到图书借阅页面
    }
}

```

## B.10 开发问题解析

在开发图书馆管理系统过程中，笔者遇到了一些问题，现在将这些问题及其解析与读者分享，希望对读者的学习有一定的帮助。

### B.10.1 如何自动计算图书归还日期

在图书馆管理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，而这个日期又不是固定不变的，它是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的，即图书归还日期=系统日期+最多借阅天数。

在本系统中是这样解决该问题的：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，最后计算归还日期。计算归还日期的方法如下：

首先取出系统时间中的“天”，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。

自动计算图书归还日期的具体代码如下：

```

//获取系统日期
Date dateU=new Date();
java.sql.Date date=new java.sql.Date(dateU.getTime());
//获取图书的最多借阅天数
String sql1="select t.days from tb_bookinfo b left join tb_booktype t on b.typeid=t.id where b.id="+bookForm.getId()+"";
ResultSet rs=conn.executeQuery(sql1);    //执行查询语句
int days=0;
try {
    if (rs.next()) {
        days = rs.getInt(1);
    }
} catch (SQLException ex) {
}
//计算归还日期
String date_str=String.valueOf(date);
String dd = date_str.substring(8,10);
String DD = date_str.substring(0,8)+String.valueOf(Integer.parseInt(dd) + days);
java.sql.Date backTime= java.sql.Date.valueOf(DD);

```

### B.10.2 如何对图书借阅信息进行统计排行

在图书馆管理系统的主界面中，提供了显示图书借阅排行榜的功能。要实现该功能，最重要的是

要知道如何获取统计排行信息，这可以通过一条 SQL 语句实现。本系统中实现对图书借阅信息进行统计排行的 SQL 语句如下：

```
select * from (SELECT bookid,count(bookid) as degree FROM tb_borrow group by bookid) as borrr join (select b.*,c.name as bookcaseName,p.pubname,t.typename from tb_bookinfo b left join tb_bookcase c on b.bookcase=c.id join tb_publishing p on b.ISBN=p.ISBN join tb_booktype t on b.typeid=t.id where b.del=0) as book on borrr.bookid=book.id order by borrr.degree desc limit 10
```

下面将对该 SQL 语句进行分析：

(1) 对图书借阅信息表进行分组并统计每本图书的借阅次数，然后使用 AS 为其指定别名为 borrr，代码如下：

```
(SELECT bookid,count(bookid) as degree FROM tb_borrow group by bookid) as borrr
```

(2) 使用左连接查询出图书的完整信息，然后使用 AS 为其指定别名为 book，代码如下：

```
(select b.*,c.name as bookcaseName,p.pubname,t.typename from tb_bookinfo b left join tb_bookcase c on b.bookcase=c.id join tb_publishing p on b.ISBN=p.ISBN join tb_booktype t on b.typeid=t.id where b.del=0) as book
```

(3) 使用 JOIN ON 语句将 borrr 和 book 连接起来，再对其按统计的借阅次数 degree 进行降序排序，并使用 LIMIT 子句限制返回的行数。