

**数据库课程设计报告**

**系 （院）：**  计算机科学学院

**专业班级：** 软工11401

**姓 名：** 王海金

**学 号：** 201404597

**指导教师： 黄岚**

**设计时间：** 2016.12.12 - 2016.12.23

**设计地点：** 12教机房

目录

[1. 设计题目 3](#_Toc471003763)

[2. 设计目的 3](#_Toc471003764)

[3. 设计任务 3](#_Toc471003765)

[4. 需求分析 3](#_Toc471003766)

[系统边界与约定 3](#_Toc471003767)

[需求概要 3](#_Toc471003768)

[4.1 读者管理 4](#_Toc471003769)

[4.2 图书管理 8](#_Toc471003770)

[4.3 借阅管理 9](#_Toc471003771)

[4.4 用户登录与用户管理 11](#_Toc471003772)

[4.5 系统功能与用户角色 12](#_Toc471003773)

[4.6 领域模型 13](#_Toc471003774)

[5. 总体设计 14](#_Toc471003775)

[5.1 MVC架构简介 14](#_Toc471003776)

[5.2 类总体设计 15](#_Toc471003777)

[6. 详细设计与实现 16](#_Toc471003778)

[6.1 数据库设计与实现 16](#_Toc471003779)

[6.2 Model层实体类设计 19](#_Toc471003780)

[6.3 Dao层类设计 30](#_Toc471003781)

[6.4view实现以及功能实现 38](#_Toc471003782)

[6. 课程设计小结 80](#_Toc471003783)

[个人小结 80](#_Toc471003784)

[指导老师意见 81](#_Toc471003785)

## 1. 设计题目

图书管理系统

## 2. 设计目的

通过对图书管理系统的系统分析、系统设计、编码和调试等工作的实践，熟悉管理信息系统的开发过程、设计方法及相关编程技术，熟练掌握数据库设计的基本理论及方法。

## 3. 设计任务

要求完成一个具有一定实用价值的图书管理系统，主要任务包括：

➀ 在Microsoft SQL Server 2000/2005/2008环境下建立图书管理系统所使用的数据库，利用企业管理器或查询分析器建立各种数据库对象，包括：数据表、视图、约束、存储过程和触发器等；

➁ 掌握JDBC编程技术，对MS SQL Server数据库进行连接和操纵；

➂ 掌握使用Java语言开发一个数据库应用系统的基本方法和步骤，熟悉一些基础功能的实现方法，如：数据维护（插删改等操作），数据查询、浏览和Excel导出，统计与报表，用户登录和权限管理等。

➃ 了解C/S或B/S应用程序的多层体系结构及三层架构方案设计思想，了解迭代式开发，熟悉面向对象设计方法及其分析与设计过程，了解UML文档及其开发过程中的作用。

## 4. 需求分析

### 系统边界与约定

* 1. 系统限定在实体书库的借阅和管理等业务范围；
  2. 不考虑图书馆的电子书库、订购、情报、人事管理等业务；
  3. 不考虑图书馆的跨区分布情况，如长江大学图书馆包括多个校区图书馆；
  4. 不考虑图书的通借通还，如长江大学读者可在湖北省高校任何一家图书馆借还图书；
  5. 不考虑珍藏图书的借阅业务；
  6. 暂不考虑与校园一卡通系统的外部接口。一卡通系统为外部系统（外部参与者），卡内记录有身份及相关信息，该系统负责身份验证工作。
  7. 期刊库和论文库的借阅和管理等业务可作为二期项目目标，视本系统使用情况而定。

### 需求概要

(1) 图书管理系统的基本功能需求包括：读者管理、图书管理、借阅管理、用户登录与用户管理等；

(2) 相关领域概念：借书证（读者）、图书、借还书记录。

(3) 系统外部参与者：读者、借书证管理员、图书管理员、借阅管理员、系统管理员。将图书馆工作人员划分为借书证管理、图书管理、借阅管理等三类人员；系统管理员负责数据库和软件系统管理，包括各类管理员用户的创建和授权、数据库备份等工作。

### 4.1 读者管理

读者管理即借书证管理，包括的业务（即用例）有：办理借书证、借书证变更、借书证挂失、解除挂失、补办借书证、注销借书证

借书证（读者）可分为2种类别：教师、学生。

借书证（教师）=借书证号、姓名、性别、所在单位、办证日期、照片等。

借书证（学生）=借书证号、学号、姓名、性别、专业、班级、办证日期、有效期、照片等。其中，有效期由学生类别决定，本科生4年、专科生3年、硕士生3年等。

相关业务规则：(1)读者凭借书证借书；(2)教师最多借书12本，借书期限最长为60天，可续借2次；学生最多借书8本，借书期限最长为30天，可续借1次；(3)处于挂失、注销状态的读者不能借书；(4)未归还图书者不能注销其借书证。

分析：增加读者类别概念，统一并简化读者信息，采用教师属性项，而学生可在所在单位填写班级，学生特有读者信息可留待以后的开发周期处理。可得到2个概念：读者、读者类别。

读者=借书证号、姓名、性别、所在单位、读者类别、办证日期、照片等。（另可加：电话、邮箱等）

读者类别=读者类别号、类别名称、可借书本数、可借书天数、可续借次数。

#### 1. 办理借书证

|  |
| --- |
| 用例名称：办理借书证 |
| 迭 代：1 （说明：第1次迭代期间的用例文档） |
| 参 与 者：借书证管理员（或系统管理员）、读者 |
| 综 述：用例起始于读者来到借书证管理办公室，管理员给读者办理新借书证。 |
| 触 发 器： （说明该用例的触发条件） |
| 前置条件：登录用户具备读者管理权限。 |
| 过程描述： （标题名或为：基本流程、基本事件流、主要成功场景等） |
| 1. 管理员输入：读者类别、所在单位、姓名。 |
| 2. 点击“查询”。 |
| 3. 系统查询读者信息，并显示查询结果。 |
| 4. 管理员判断该读者是否已办理借书证，若是，则可取消办证过程。（人工操作） |
| 5. 管理员输入读者信息：读者类别、姓名、性别、所在单位、办证日期、照片等。（类别、姓名、单位等可预置为过程1中输入值，办证日期可预置为系统日期） |
| 6. 确认办证。 |
| 后置条件：创建新的读者对象，并具有新的借书证号；系统记录读者信息。 |
| 业务规则：借书证号应具有唯一性。 |
| 待解决问题： |
| 1. 打印借书证； |
| 注 释：未考虑校园一卡通做为借书证使用的情况 |
| 作者与日期： 2013-7-24日，LSH，版本号1.0 |

#### 2. 变更借书证

变更借书证类别、有效期、所在单位等，以及其它信息；借书证号不可更改。

用例

|  |
| --- |
| 用例名称：变更借书证 |
| 参与者：借书证管理员、读者 |
| 综述: 用例起始于读者来到借书证管理办公室，管理员给读者办理借书证挂失。 |
| 前置条件：登录用户具备读者管理权限，且读者已经存在 |
| 过程描述： |
| 1. 输入借书证号，或输入读者的类别、单位、姓名。 |
| 2.点击“查询”。 |
| 3系统查询并显示读者信息。 |
| 4. 找到其借书证，变更借书证数据 |
| 5.确认更改  后置条件：修改读者信息，系统记录新数据  业务规则：读者使用新数据进行操作 |

#### 3. 挂失借书证

|  |
| --- |
| 用例名称：挂失借书证 |
| 参 与 者：借书证管理员、读者 |
| 综 述：用例起始于读者来到借书证管理办公室，管理员给读者办理借书证挂失。 |
| 前置条件：登录用户具备读者管理权限，且读者已经存在 |
| 过程描述： |
| 1. 输入借书证号，或输入读者的类别、单位、姓名。 |
| 2. 点击“查询”。 |
| 3. 系统查询并显示读者信息。 |
| 4. 找到其借书证，且读者状态为有效。 |
| 5. 确认挂失。 |
| 后置条件：修改读者状态为挂失；系统记录读者信息。 |
| 业务规则：借书证挂失后禁止读者借书。 |

#### 4. 解除挂失

|  |
| --- |
| 用例名称：解除挂失 |
| 参 与 者：借书证管理员、读者 |
| 综 述：用例起始于读者来到借书证管理办公室，管理员给读者办理解除借书证挂失。 |
| 前置条件：登录用户具备读者管理权限，且读者已经存在 |
| 过程描述： |
| 1. 输入借书证号，或输入读者的类别、单位、姓名。 |
| 2. 点击“查询”。 |
| 3. 系统查询并显示读者信息。 |
| 4. 找到其借书证，且读者状态为挂失。 |
| 5. 确认解除挂失。 |
| 后置条件：修改读者状态为有效；系统记录读者信息。 |

#### 5. 补办借书证

补办借书证，办理新借书证（新借书证号，其它信息不变），并将原借书证的借阅记录转移到新证上，原借书证注销。

用例

|  |
| --- |
| 用例名称：办理借书证 |
| 迭 代：1 （说明：第1次迭代期间的用例文档） |
| 参 与 者：借书证管理员（或系统管理员）、读者 |
| 综 述：用例起始于读者来到借书证管理办公室，管理员给读者办理新借书证。 |
| 触 发 器： （说明该用例的触发条件） |
| 前置条件：登录用户具备读者管理权限。 |
| 过程描述： （标题名或为：基本流程、基本事件流、主要成功场景等） |
| 1. 管理员输入：读者类别、所在单位、姓名。 |
| 2. 点击“查询”。 |
| 3. 系统查询读者信息，并显示查询结果。 |
| 4. 管理员判断该读者是否已办理借书证，若是，则可补办。 |
| 5. 管理员将读者证的状态改为有效 |
| 6. 确认办证。 |
| 后置条件：创建的读者对象，借书证号不变；系统记录读者信息。 |
| 业务规则：借书证号应具有唯一性。 |

#### 6. 注销借书证

用例

|  |
| --- |
| 用例名称：注销借书证 |
| 参 与 者：借书证管理员、读者 |
| 综 述：用例起始于读者来到借书证管理办公室，管理员给读者办理借书证注销。 |
| 前置条件：登录用户具备读者管理权限，且读者已经存在 |
| 过程描述： |
| 1. 输入借书证号，或输入读者的类别、单位、姓名。 |
| 2. 点击“查询”。 |
| 3. 系统查询并显示读者信息。 |
| 4. 找到其借书证，且读者状态为有效。 |
| 5. 确认注销。 |
| 后置条件：修改读者状态为注销；系统记录读者信息。 |
| 业务规则：借书证注销后禁止读者借书。 |

#### 7. 读者类型管理

从系统的功能完整性来看，还存在一个隐性的功能需求：读者类型管理，即读者类型信息的插、删、改、查等数据维护操作。

用例

|  |
| --- |
| 用例名称：读者类型管理 |
| 综述：读者管理员插、删、改、查读者类型 |
| 前置条件：登录用户具备读者类型管理权限 |
| 1. 插入读者类型   点击添加，输入数据，点击完成 |
| 1. 更改读者类型   将数据更改，点击更新 |
| 1. 删除读者类型   点击删除 |
| 后置条件：读者类型的更改将在页面显示出来 |
| 规则：添加后，读者的类型也会变化 |

### 4.2 图书管理

包括业务（用例）：图书编目、新书入库、图书信息维护、图书变卖与销毁处理\*等。

图书信息=书号、书名、作者、出版社、出版日期、ISBN、分类号、语言、页数、单价、内容简介、图书封面、图书状态等；（图书状态包括：在馆、借出、遗失、变卖、销毁）

#### 1. 图书编目

新书入库前，图书管理员对图书进行图书编目工作，即根据该馆的现有图书分类目录信息编写新书的分类号，需要时根据图书分类法增加新的图书分类条目。

图书信息= … 、分类号、…

图书分类目录=分类号、分类名。图书分类信息主要用于统计，如中文图书分类条目：

-----------------------------------------------------------

分类号 分类名

TP 自动化技术、计算机技术

TP3 计算技术、计算机技术

TP31 计算机软件

TP311 程序设计、软件工程

TP312 程序语言、算法语言

TP311.13 数据库理论与系统

TP311.131 数据库理论

新书分类号，如图书“C#高级编程(第7版)”可编为：TP312/3033.2

用例名称：图书编目（略）

#### 2. 新书入库

|  |
| --- |
| 用例名称：新书入库 |
| 参 与 者：图书管理员 |
| 前置条件：登录用户具备图书管理身份和权限 |
| 过程描述： |
| 1. 输入：图书信息(书名、作者、出版社、出版日期、语言、页数、单价、ISBN、分类号、内容简介等)、入库数量(N，N>=1)、入馆日期。 |
| 3. 确认图书入库。 |
| 4. 图书馆工作人员对图书进行贴书签（分类号）、印书号（或条形码）、盖图书馆印章。 |
| 5. 图书上架。 |
| 后置条件：创建N个图书对象，书号分别为BkNO、BkNO+1、…、BkNO+N-1，这N本图书的状态均修改为在馆；系统记录图书信息。 |
| 注 释：BkNO+N可作为下一次新书入库的起始书号。 |

#### 3. 图书信息维护

图书信息的查、删、改等信息维护操作。

用例名称：图书信息维护

|  |
| --- |
| 用例名称：图书信息维护 |
| 参与者：图书管理员 |
| 前置条件：登录用户具备图书管理身份和权限 |
| 过程描述： |
| 1.输入信息，使用简单查找或高级查找查找书籍 |
| 2.点击维护书籍，转入维护书籍页面 |
| 3.更新图书数据 |
| 4.确认更新 |
| 后置条件：更新全部图书信息 |

### 4.3 借阅管理

包括业务用例：借书、续借、还书等。还书过程涉及超期罚款、遗失图书罚款等业务规则。

罚款规则：（1）超期罚款规则 应罚款金额=超期天数\*罚款率，罚款率=0.05元/天，罚款率可能随时间或读者类别而变化；实际罚款金额<=应罚款金额，根据实际情况可以进行减免。（2）遗失罚款规则 遗失图书应罚款金额=3\*图书单价；实际罚款金额在（1\*图书单价，3\*图书单价）之间。（3）遗失罚款规则优先于超期罚款规则。

借书记录=借书证号、书号、借书操作员、借书日期、应还日期

续借记录=借书证号、书号、续借操作员、续借日期、应还日期，续借次数

还书记录=借书证号、书号、还书操作员、还书日期、应还日期，超期天数、应罚款金额，实际罚款金额

分析：借书记录、续借记录、还书记录等信息可合并为借阅记录，续借可舍去部分信息。借阅历史记录应该长期保存在数据库中，以便于统计分析，另外，读者对一本书可能多次借阅，故借阅信息应该增加一个标识项：借书顺序号。

借阅信息=借书顺序号、借书证号、书号、借书操作员、借书日期、应还日期，续借次数、还书操作员、还书日期，超期天数、应罚款金额，实际罚款金额

#### 1. 借书

|  |
| --- |
| 用例名称：借书 |
| 参 与 者：借阅管理员、读者 |
| 综 述：用例起始于读者带着图书来到图书借阅大厅，管理员给读者办理借书手续。 |
| 前置条件：登录用户具备借阅管理权限。 |
| 过程描述： |
| 1. 管理员输入借书证号，点击“查询读者”。 |
| 1. 系统查询读者信息、未归还图书信息（含超期），并进行显示。 |
| 1. 系统判断读者可否借书（借书证状态为有效，已借书数量小于可借书数量，不存在超期未归还图书）。若不可借书，则禁止借书。 |
| 1. 管理员输入待借图书的书号，点击“查询图书”。 |
| 1. 系统查询，并显示图书信息。 |
| 1. 系统判断图书是否在馆，若不在馆，则禁止借书。 |
| 1. 管理员点击“借书”。 |
| 后置条件：创建借阅记录对象（借书顺序号由系统自动产生，借书证号和书号为上述输入值，借书操作员=登录用户，借书日期=系统日期，应还日期=系统日期+可借书天数，续借次数=0，还书日期=NULL，还书操作员=NULL），并标记为未归还；修改读者对象的已借书数量+1；修改图书状态为借出；系统记录借阅对象、读者对象、图书对象。 |
| 业务规则：(1)借书证状态为挂失、注销者不能借书；(2)借书数量不能超过可借书数量；(3)有超期未归还图书者不能借书。 |
| 注 释：未归还图书状态可采用还书日期为空（NULL）来标记，或另加属性表示是否还书。 |

#### 2. 续借

|  |
| --- |
| 用例名称：续借 |
| 参 与 者：借阅管理员、读者 |
| 前置条件：登录用户具备借阅管理权限、图书状态为借出 |
| 过程描述： |
| 1. 输入待续借图书书号。 |
| 1. 系统查询并获取图书对象、未归还状态的借阅记录对象、及其读者对象，并进行显示。 |
| 1. 系统判断可否续借（续借次数<可续借次数, 读者状态为有效）。 |
| 1. 点击“续借”。 |
| 后置条件：修改借阅记录对象（续借次数+1，应还日期+=可借书天数），图书状态为借出，并保存到数据库中。 |
| 业务规则：(1)借书证状态为挂失、注销者不能续借；(2)续借次数不能超过可续借次数。 |
| 注 释：续借前后图书状态应为借出 |

#### 3. 还书

还书用例需要处理超期罚款、遗失图书罚款处理2种情况。

用例名称：归还图书

|  |
| --- |
| 用例名称：还书 |
| 参与者：借阅管理员、读者 |
| 前置条件：登录用户具备借阅管理权限、读者借阅了图书 |
| 过程描述 |
| 1.输入读者借书号  2.获取读者未书籍列表 |
| 3.查看是否超期，并是否有欠款，如无，允许还书 |
| 4.点击还书 |
| 后置条件：读者借书量减一，图书状态改为在馆 |

### 4.4 用户登录与用户管理

包括用例：用户登录、密码修改、用户管理\*，为本系统的基础和主要功能。

用户包括2类：读者、管理员。其中，管理员用户权限是4种角色的组合：借书证管理、图书管理、借阅管理、系统管理；系统管理员负责所有管理员用户及其权限的管理，借书证管理员负责读者管理（即借书证管理）。

管理员是读者，但读者不一定是管理员；读者与管理员间存在(1对0..1)联系。

读者信息+=密码。

管理员信息=用户号、用户名、密码、管理角色

管理角色设计：0-系统管理，1-读者管理，2-图书管理，3-借阅管理，4-读者。

读者信息+=密码、管理角色。需修改用例办理借书证。

#### 1. 用户登录

|  |
| --- |
| 用例名称：用户登录 |
| 参 与 者：读者或管理员 |
| 前置条件：无 |
| 过程描述： |
| 1. 输入用户号（即借书证号）、密码；密码采用掩码“\*”号显示。 |
| 2. 点击“登录”。 |
| 3. 系统根据借书证号获取读者对象。 |
| 3a. 未获取读者对象时，显示无此用户，转入1； |
| 3b. 密码未匹配时，显示密码错误，并清空密码，转入1； |
| 4. 进入系统主程序界面。 |
| 后置条件：根据登录用户权限显示或隐藏相应的系统功能（菜单等）。 |
| 待解决问题： |
| 1. 挂失或注销借书证者不能登录。 |
| 注 释：该登录用户（读者）对象应为全局可见 |

系统功能与用户角色之间的关系参见“表1 系统功能与用户角色关系表”。

#### 2. 密码修改

|  |
| --- |
| 用例名称：修改密码 |
| 参 与 者：用户 |
| 综 述：用户登录后修改自己的登录密码。 |
| ..（略） |

#### 3. 用户管理

|  |
| --- |
| 用例名称：用户管理 |
| 参 与 者：系统管理员、图书馆工作人员 |
| 综 述：系统管理员对图书馆工作人员进行权限管理。 |
| 前置条件：登录用户为系统管理员，图书馆工作人员需先办理借书证 |
| 过程描述： |
| 1. 输入借书证号，或输入读者的类别、单位、姓名。 |
| 2. 点击“查询”。 |
| 3. 系统查询，并显示读者信息。 |
| 4. 找到借书证后，设置该读者的管理角色。 |
| 5. 确认完成。 |
| 后置条件：修改读者的管理员角色，系统记录读者信息。 |
| 注 释： 管理角色有4种：借书证管理、图书管理、借阅管理、系统管理 |

### 4.5 系统功能与用户角色

在读者信息表中使用rdAdminRoles来存储使用者的权限，0为系统管理员，1为读者管理员，2为图书管理员，3为借阅管理员，4为读者

表 1 系统功能与用户角色关系表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **系统功能** | **借书证管理** | **图书管理** | **借阅管理** | **系统管理** | **读者** |
| 1 | 读者管理 | √ |  |  | （√） |  |
| 2 | 图书管理 |  | √ |  | √ |  |
| 3 | 借阅管理 |  |  | √ | √ |  |
| 4 | 用户登录 | √ | √ | √ | √ | √ |
| 5 | 密码修改 | √ | √ | √ | √ | √ |
| 6 | 用户管理 |  |  |  | √ |  |
| 7 | 图书查询 | √ | √ | √ | √ | √ |

### 4.6 领域模型

#### 领域概念小结

读者=读者号（借书证号）、姓名、性别、所在单位、电话、邮箱地址、办证日期、照片、借书证状态、已借书数量、密码、管理角色。（借书证状态：有效、挂失、注销）

读者类别=读者类别号、读者类别名称、可借书数量、可借书天数、可续借次数，罚款率，证件有效期。

图书=书号、书名、作者、出版社、出版日期、ISBN、分类号、语言、页数、单价、内容简介、图书封面、图书状态。（图书状态包括：在馆、借出、遗失、销毁、卖出）

借阅信息=借书顺序号、读者号、书号、借书操作员、借书日期、应还日期、续借次数、还书操作员、还书日期，超期天数、应罚款金额、实际罚款金额。

#### 领域模型

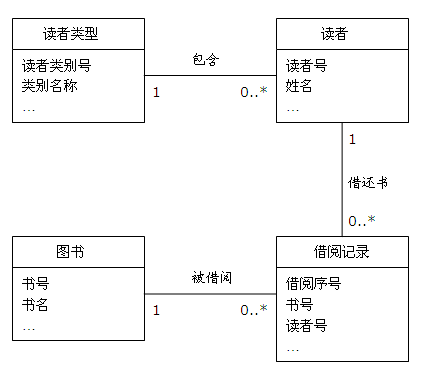


图 1 图书管理系统概念模型(V1)

## 5. 总体设计

### 5.1 MVC架构简介

MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化

用户界面的结构中。



图 1 MVC架构示意图

**Model（模型）**是应用程序中用于处理应用程序数据逻辑的部分。  
　　通常模型对象负责在数据库中存取数据。

**View（视图）**是应用程序中处理数据显示的部分。  
　　通常视图是依据模型数据创建的。

**Controller（控制器）**是应用程序中处理用户交互的部分。  
　　通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据

### 5.2 类总体设计

Model、View、Controller的初步设计如表 2。

表 2 Model、View、Controller的类设计

|  |  |  |
| --- | --- | --- |
| 层 | 类名 | 说明 |
| Model | TB\_ReaderType | 读者类型实体类 |
| TB\_ReaderTypeDao | 读者类型访问数据库方法 |
| TB\_Reader | 读者实体类 |
| TB\_ReaderDao | 读者数据库操作 |
| TB\_Book | 图书实体类 |
| TB\_BookDao | 图书数据库操作 |
| TB\_Borrow | 借阅记录实体类 |
| TB\_BorrowDao | 借阅记录数据库操作 |
| View | login.jsp | 登录界面 |
| Main.jsp | 主界面 |
| findBook.jsp | 书籍查找 |
| myPassword.jsp | 密码更改 |
| findReader.jsp | 密码重置 |
| insertReader.jsp | 新生入馆 |
| findReader.jsp | 查找读者 |
| findReaderType.jsp | 读者类型 |
| password.jsp | 密码重置 |
| insertBook.jsp | 新书入库 |
| findBook.jsp | 查找书籍 |
| borrowBook.jsp | 借阅管理 |
| adminRoles.jsp | 权限管理 |
| Controller | FindAdminRolesServlet | 权限管理 |
| FindBookServlet | 图书管理 |
| InsertBookServlet |
| UpdateBookServlet |
| ShowUpdateReaderServlet |
| FindReaderServlet | 读者管理 |
| InsertReaderServlet |
| UpdateReaderServlet |
| ShowUpdateReaderServlet |
| FindReaderServlet | 读者类型管理 |
|  |
| BorrowBookServlet | 借阅管理 |
| BorrowFindBookServlet |
| BorrowFindReaderServlet |
| NextBorrowBookServlet |
| ReturnBookServlet | 借阅管理 |
| UpdatePasswordServlet | 密码管理 |

## 6. 详细设计与实现

### 6.1 数据库设计与实现

#### 读者类别表(TB\_ReaderType)

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 说明 |
| 1 | rdType | SmallInt | 读者类别【主键】 |
| 2 | rdTypeName | Nvarchar(20) | 读者类别名称【唯一、非空】 |
| 3 | CanLendQty | Int | 可借书数量 |
| 4 | CanLendDay | Int | 可借书天数 |
| 5 | CanContinueTimes | Int | 可续借的次数 |
| 6 | PunishRate | Float | 罚款率（元/天） |
| 7 | DateValid | SmallInt | 证书有效期（年）【非空，0表示永久有效】 |

#### 读者信息表(TB\_Reader)

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 说明 |
| 1 | rdID | Int | 读者编号/借书证号【主键】 |
| 2 | rdName | nvarchar(20) | 读者姓名 |
| 3 | rdSex | nchar(1) | 性别，男/女 |
| 4 | rdType | SmallInt | 读者类别【外键TB\_ReaderType】【非空】 |
| 5 | rdDept | nvarchar (20) | 单位代码/单位名称 |
| 6 | rdPhone | nvarchar(25) | 电话号码 |
| 7 | rdEmail | nvarchar(25) | 电子邮箱 |
| 8 | rdDateReg | datetime | 读者登记日期/办证日期 |
| 9 | rdPhoto | image | 读者照片 |
| 10 | rdStatus | nchar(2) | 证件状态，3个：有效、挂失、注销 |
| 11 | rdBorrowQty | Int | 已借书数量(缺省值0) |
| 12 | rdPwd | nvarchar (20) | 读者密码(初值123)，可加密存储 |
| 13 | rdAdminRoles | SmallInt | 管理角色，0-读者、1-借书证管理、2-图书管理、4-借阅管理、8-系统管理，可组合 |

#### 图书信息表(TB\_Book)

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 说明 |
| 1 | bkID | Int | 图书序号【标识列，主键】 |
| 2 | bkCode | Nvarchar (20) | 图书编号或条码号（前文中的书号） |
| 3 | bkName | Nvarchar(50) | 书名 |
| 4 | bkAuthor | Nvarchar(30) | 作者 |
| 5 | bkPress | Nvarchar(50) | 出版社 |
| 6 | bkDatePress | datetime | 出版日期 |
| 7 | bkISBN | Nvarchar (15) | ISBN书号 |
| 8 | bkCatalog | Nvarchar(30) | 分类号（如：TP316-21/123） |
| 9 | bkLanguage | SmallInt | 语言，0-中文，1-英文，2-日文，3-俄文，  4-德文，5-法文 |
| 10 | bkPages | Int | 页数 |
| 11 | bkPrice | Money | 价格 |
| 12 | bkDateIn | DateTime | 入馆日期 |
| 13 | bkBrief | Text | 内容简介 |
| 14 | bkCover | image | 图书封面照片 |
| 15 | bkStatus | NChar(2) | 图书状态，在馆、借出、遗失、变卖、销毁 |

说明：bkCode为前文中的书号，这里没有设计为关键字，而增加bkID字段作为表关键字，其原因为：防止新书入库时起始书号输入错误时，因关键字而不易修改。

#### 借阅信息表(TB\_Borrow)

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 说明 |
| 1 | BorrowID | Numeric(12,0) | 借书顺序号【主键】 |
| 2 | rdID | Int | 读者序号【外键TB\_Reader】 |
| 3 | bkID | Int | 图书序号【外键TB\_Book】 |
| 4 | ldContinueTimes | Int | 续借次数（第一次借时，记为0） |
| 5 | ldDateOut | DateTime | 借书日期 |
| 6 | ldDateRetPlan | DateTime | 应还日期 |
| 7 | ldDateRetAct | DateTime | 实际还书日期 |
| 8 | ldOverDay | Int | 超期天数 |
| 9 | ldOverMoney | Money | 超期金额（应罚款金额） |
| 10 | ldPunishMoney | Money | 罚款金额 |
| 11 | lsHasReturn | Bit | 是否已经还书，缺省为0-未还 |
| 12 | OperatorLend | Nvarchar(20) | 借书操作员 |
| 13 | OperatorRet | Nvarchar(20) | 还书操作员 |

#### 数据库实现（具体见library.sql,此处为框架）

create table TB\_ReaderType --读者类别

(

rdType smallInt primary key,--读者类别

rdTypeName nvarchar(20) constraint uq\_tn unique not null,--类别名称

canLendQty int ,--可借书数量

canLendDay int , --可借书天数

canContinueTimes int,--可续借次数

punishRate Float,--罚款率(元/天)

dateValid SmallInt ,--证书有效期(年)

)

create table TB\_Reader --读者信息

(

rdID int primary key,--读者编号

rdName nvarchar(20),--姓名

rdSex nchar(1) check(rdSex in('男','女')),--性别

rdType SmallInt foreign key references TB\_ReaderType(rdType) not null ,--类别

rdDept nvarchar(20),--单位名称

rdPhone nvarchar(25),--电话号码

rdEmail nvarchar(25),--电子邮箱

rdDateReg datetime ,--读者登记日期

rdPhoto nvarchar(25),--照片

rdStatus nchar(2) check(rdStatus in ('有效','挂失','注销')),--证件状态

rdBorrowQty int default 0,--已借书数量

rdPwd nvarchar(20) default '123',--读者密码

rdAdminRoles smallInt --管理角色--0,全部--1，读者管理--2，图书管理--3，借书管理--4，读者

)

create table TB\_Book

(

bkID int primary key,--图书序号

bkCode nvarchar(20),--图书编号

bkName nvarchar(50),--书名

bkAuthor nvarchar(30),--作者

bkPress nvarchar(50),--出版社

bkDatePress datetime,--出版日期

bkISBN nvarchar(15),--ISBN书号

bkCatalog nvarchar(30),--分类号

bkLanguage smallint ,--语言,0-中文，1-英文，2-日文，3-俄文，4-德文，5-法文

bkPages int ,--页数

bkPrice money,--价格

bkDateIn datetime ,--入馆日期

bkBrief text ,--内容简介

bkCover nvarchar(25),--图书封面照片

bkStatus nchar(2) check(bkStatus in('在馆','借出','遗失','变卖','销毁'))

)

create table TB\_Borrow

(

borrowID numeric(12,0) primary key,--借书顺序号

rdID int foreign key references TB\_Reader(rdID),--读者序号

bkID int foreign key references TB\_Book(bkID),--图书序号

idContinueTimes int ,--续借次数

idDateOut datetime ,--借书日期

idDateRetPlan datetime,--应还日期

idDateRetAct datetime,--实际还书日期

idOverDay int ,--超期天数

idOverMoney money ,--应罚款金额

idPunishMoney money,--罚款金额

isHasReturn bit default 0,--是否已经还书,0-未还

operatorLend nvarchar(20),--借书操作员

operatorRet nvarchar(20)--还书操作员

)

insert into [TB\_ReaderType] values(10,'教师',12,60,2,0.05,0);

insert into [TB\_ReaderType] values(20,'本科生',8,30,1,0.05,4);

insert into [TB\_ReaderType] values(21,'专科生',8,30,1,0.05,3);

insert into [TB\_ReaderType] values(30,'硕士研究生',8,30,1,0.05,3);

insert into [TB\_ReaderType] values(31,'博士研究生',8,30,1,0.05,4);

GO

### 6.2 Model层实体类设计

在com.library.model包中添加4个实体类：TB\_ReaderType、TB\_Reader、TB\_Book、TB\_Borrow，分别对应4个数据库表。实体类的属性与数据库表结构保持一致（名称、类型）；实体类应尽量简单，除了实体对象的复制与比较等方法外，不宜添加过多方法。

#### TB\_ReaderType类

TB\_ReaderType类的实现代码如下：

**package** com.library.model;

//读者类别

**public** **class** TB\_ReaderType {

**private** **int** rdType;//读者类别

**private** String rdTypeName;//类别名称

**private** **int** canLendQty;//可借书数量

**private** **int** canLendDay;//可借书天数

**private** **int** canContinueTimes;//可续借次数

**private** **float** punishRate;//罚款率

**private** **int** dateValid;//证书有效期

**public** **int** getRdType() {

**return** rdType;

}

**public** **void** setRdType(**int** rdType) {

**this**.rdType = rdType;

}

**public** String getRdTypeName() {

**return** rdTypeName;

}

**public** **void** setRdTypeName(String rdTypeName) {

**this**.rdTypeName = rdTypeName;

}

**public** **int** getCanLendQty() {

**return** canLendQty;

}

**public** **void** setCanLendQty(**int** canLendQty) {

**this**.canLendQty = canLendQty;

}

**public** **int** getCanLendDay() {

**return** canLendDay;

}

**public** **void** setCanLendDay(**int** canLendDay) {

**this**.canLendDay = canLendDay;

}

**public** **int** getCanContinueTimes() {

**return** canContinueTimes;

}

**public** **void** setCanContinueTimes(**int** canContinueTimes) {

**this**.canContinueTimes = canContinueTimes;

}

**public** **float** getPunishRate() {

**return** punishRate;

}

**public** **void** setPunishRate(**float** punishRate) {

**this**.punishRate = punishRate;

}

**public** **int** getDateValid() {

**return** dateValid;

}

**public** **void** setDateValid(**int** dateValid) {

**this**.dateValid = dateValid;

}

}

添加复制构造函数为：

**public** TB\_ReaderType(**int** rdType, String rdTypeName, **int** canLendQty, **int** canLendDay, **int** canContinueTimes,

**float** punishRate, **int** dateValid) {

**super**();

**this**.rdType = rdType;

**this**.rdTypeName = rdTypeName;

**this**.canLendQty = canLendQty;

**this**.canLendDay = canLendDay;

**this**.canContinueTimes = canContinueTimes;

**this**.punishRate = punishRate;

**this**.dateValid = dateValid;

}

#### TB\_ Reader类

**package** com.library.model;

//读者类别

**public** **class** TB\_ReaderType {

**private** **int** rdType;//读者类别

**private** String rdTypeName;//类别名称

**private** **int** canLendQty;//可借书数量

**private** **int** canLendDay;//可借书天数

**private** **int** canContinueTimes;//可续借次数

**private** **float** punishRate;//罚款率

**private** **int** dateValid;//证书有效期

**public** **int** getRdType() {

**return** rdType;

}

**public** **void** setRdType(**int** rdType) {

**this**.rdType = rdType;

}

**public** String getRdTypeName() {

**return** rdTypeName;

}

**public** **void** setRdTypeName(String rdTypeName) {

**this**.rdTypeName = rdTypeName;

}

**public** **int** getCanLendQty() {

**return** canLendQty;

}

**public** **void** setCanLendQty(**int** canLendQty) {

**this**.canLendQty = canLendQty;

}

**public** **int** getCanLendDay() {

**return** canLendDay;

}

**public** **void** setCanLendDay(**int** canLendDay) {

**this**.canLendDay = canLendDay;

}

**public** **int** getCanContinueTimes() {

**return** canContinueTimes;

}

**public** **void** setCanContinueTimes(**int** canContinueTimes) {

**this**.canContinueTimes = canContinueTimes;

}

**public** **float** getPunishRate() {

**return** punishRate;

}

**public** **void** setPunishRate(**float** punishRate) {

**this**.punishRate = punishRate;

}

**public** **int** getDateValid() {

**return** dateValid;

}

**public** **void** setDateValid(**int** dateValid) {

**this**.dateValid = dateValid;

}

}

添加复制构造函数：

**public** TB\_ReaderType(**int** rdType, String rdTypeName, **int** canLendQty, **int** canLendDay, **int** canContinueTimes,

**float** punishRate, **int** dateValid) {

**super**();

**this**.rdType = rdType;

**this**.rdTypeName = rdTypeName;

**this**.canLendQty = canLendQty;

**this**.canLendDay = canLendDay;

**this**.canContinueTimes = canContinueTimes;

**this**.punishRate = punishRate;

**this**.dateValid = dateValid;

}

#### TB\_Book类

package com.library.model;

//图书信息

import java.math.BigDecimal;

import java.sql.Date;

public class TB\_Book {

private int bkId;//图书序号

private String bkCode;//图书编号

private String bkName;//书名

private String bkAuthor;//作者

private String bkPress;//出版社

private Date bkDatePress;//出版日期

private String bkISBN;//ISBN书号

private String bkCatalog;//分类号

private int bkLanguage;//语言

private int bkPages;//页数

private BigDecimal bkPrice;//价格

private Date bkDateIn;//入馆日期

private String bkBrief;//内容介绍

private String bkCover;//图书封面照片

private String bkStatus;//图书状态

public String getBkCode() {

return bkCode;

}

public void setBkCode(String bkCode) {

this.bkCode = bkCode;

}

public String getBkName() {

return bkName;

}

public void setBkName(String bkName) {

this.bkName = bkName;

}

public String getBkAuthor() {

return bkAuthor;

}

public void setBkAuthor(String bkAuthor) {

this.bkAuthor = bkAuthor;

}

public String getBkPress() {

return bkPress;

}

public void setBkPress(String bkPress) {

this.bkPress = bkPress;

}

public Date getBkDatePress() {

return bkDatePress;

}

public void setBkDatePress(Date bkDatePress) {

this.bkDatePress = bkDatePress;

}

public String getBkISBN() {

return bkISBN;

}

public void setBkISBN(String bkISBN) {

this.bkISBN = bkISBN;

}

public String getBkCatalog() {

return bkCatalog;

}

public void setBkCatalog(String bkCatalog) {

this.bkCatalog = bkCatalog;

}

public int getBkLanguage() {

return bkLanguage;

}

public void setBkLanguage(int bkLanguage) {

this.bkLanguage = bkLanguage;

}

public int getBkPages() {

return bkPages;

}

public void setBkPages(int bkPages) {

this.bkPages = bkPages;

}

public BigDecimal getBkPrice() {

return bkPrice;

}

public void setBkPrice(BigDecimal bkPrice) {

this.bkPrice = bkPrice;

}

public Date getBkDateIn() {

return bkDateIn;

}

public void setBkDateIn(Date bkDateIn) {

this.bkDateIn = bkDateIn;

}

public String getBkBrief() {

return bkBrief;

}

public void setBkBrief(String bkBrief) {

this.bkBrief = bkBrief;

}

public String getBkCover() {

return bkCover;

}

public void setBkCover(String bkCover) {

this.bkCover = bkCover;

}

public String getBkStatus() {

return bkStatus;

}

public void setBkStatus(String bkStatus) {

this.bkStatus = bkStatus;

}

public int getBkId() {

return bkId;

}

public void setBkId(int bkId) {

this.bkId = bkId;

}

}

添加复制构造函数：

public TB\_Book(int bkId, String bkCode, String bkName, String bkAuthor, String bkPress, Date bkDatePress,

String bkISBN, String bkCatalog, int bkLanguage, int bkPages, BigDecimal bkPrice, Date bkDateIn,

String bkBrief, String bkCover, String bkStatus) {

super();

this.bkId = bkId;

this.bkCode = bkCode;

this.bkName = bkName;

this.bkAuthor = bkAuthor;

this.bkPress = bkPress;

this.bkDatePress = bkDatePress;

this.bkISBN = bkISBN;

this.bkCatalog = bkCatalog;

this.bkLanguage = bkLanguage;

this.bkPages = bkPages;

this.bkPrice = bkPrice;

this.bkDateIn = bkDateIn;

this.bkBrief = bkBrief;

this.bkCover = bkCover;

this.bkStatus = bkStatus;

}

public TB\_Book(String bkCode, String bkName, String bkAuthor, String bkPress, Date bkDatePress, String bkISBN,

String bkCatalog, int bkLanguage, int bkPages, BigDecimal bkPrice, Date bkDateIn, String bkBrief,

String bkCover, String bkStatus) {

super();

this.bkCode = bkCode;

this.bkName = bkName;

this.bkAuthor = bkAuthor;

this.bkPress = bkPress;

this.bkDatePress = bkDatePress;

this.bkISBN = bkISBN;

this.bkCatalog = bkCatalog;

this.bkLanguage = bkLanguage;

this.bkPages = bkPages;

this.bkPrice = bkPrice;

this.bkDateIn = bkDateIn;

this.bkBrief = bkBrief;

this.bkCover = bkCover;

this.bkStatus = bkStatus;

}

#### TB\_Borrow类

package com.library.model;

//借书信息

import java.math.BigDecimal;

import java.sql.Date;

public class TB\_Borrow {

// private BigDecimal borrowID;//借书顺序号

private int rdID;//读者序号

private int bkID;//图书序号

private int idContinueTimes;//续借次数

private Date idDateOut;//借书日期

private Date idDateRetPlan;//应还日期

private Date idDateRetAct;//实际还书日期

private int idOverDay;//超期天数

private BigDecimal idOverMoney;//应罚款金额

private BigDecimal idPunishMoney;//罚款金额

private boolean isHasReturn;//是否还书

private String operatorLend;//借书操作员

private String operatorRet;//还书操作员

public int getRdID() {

return rdID;

}

public void setRdID(int rdID) {

this.rdID = rdID;

}

public int getBkID() {

return bkID;

}

public void setBkID(int bkID) {

this.bkID = bkID;

}

public int getIdContinueTimes() {

return idContinueTimes;

}

public void setIdContinueTimes(int idContinueTimes) {

this.idContinueTimes = idContinueTimes;

}

public Date getIdDateOut() {

return idDateOut;

}

public void setIdDateOut(Date idDateOut) {

this.idDateOut = idDateOut;

}

public Date getIdDateRetPlan() {

return idDateRetPlan;

}

public void setIdDateRetPlan(Date idDateRetPlan) {

this.idDateRetPlan = idDateRetPlan;

}

public Date getIdDateRetAct() {

return idDateRetAct;

}

public void setIdDateRetAct(Date idDateRetAct) {

this.idDateRetAct = idDateRetAct;

}

public int getIdOverDay() {

return idOverDay;

}

public void setIdOverDay(int idOverDay) {

this.idOverDay = idOverDay;

}

public BigDecimal getIdOverMoney() {

return idOverMoney;

}

public void setIdOverMoney(BigDecimal idOverMoney) {

this.idOverMoney = idOverMoney;

}

public BigDecimal getIdPunishMoney() {

return idPunishMoney;

}

public void setIdPunishMoney(BigDecimal idPunishMoney) {

this.idPunishMoney = idPunishMoney;

}

public boolean isHasReturn() {

return isHasReturn;

}

public void setHasReturn(boolean isHasReturn) {

this.isHasReturn = isHasReturn;

}

public String getOperatorLend() {

return operatorLend;

}

public void setOperatorLend(String operatorLend) {

this.operatorLend = operatorLend;

}

public String getOperatorRet() {

return operatorRet;

}

public void setOperatorRet(String operatorRet) {

this.operatorRet = operatorRet;

}

}

添加复制构造函数

public TB\_Borrow(int rdID, int bkID, int idContinueTimes, Date idDateOut, Date idDateRetPlan, Date idDateRetAct,

int idOverDay, BigDecimal idOverMoney, BigDecimal idPunishMoney,boolean isHasReturn, String operatorLend,

String operatorRet) {

super();

this.rdID = rdID;

this.bkID = bkID;

this.idContinueTimes = idContinueTimes;

this.idDateOut = idDateOut;

this.idDateRetPlan = idDateRetPlan;

this.idDateRetAct = idDateRetAct;

this.idOverDay = idOverDay;

this.idOverMoney = idOverMoney;

this.idPunishMoney = idPunishMoney;

this.isHasReturn = isHasReturn;

this.operatorLend = operatorLend;

this.operatorRet = operatorRet;

}

public TB\_Borrow(int rdID, int bkID, int idContinueTimes, Date idDateOut, Date idDateRetPlan,boolean isHasReturn,String operatorLend) {

super();

this.rdID = rdID;

this.bkID = bkID;

this.idContinueTimes = idContinueTimes;

this.idDateOut = idDateOut;

this.idDateRetPlan = idDateRetPlan;

this.isHasReturn = isHasReturn;

this.operatorLend = operatorLend;

}

public TB\_Borrow(int rdID, int bkID, int idContinueTimes, Date idDateOut, Date idDateRetPlan,boolean isHasReturn) {

super();

this.rdID = rdID;

this.bkID = bkID;

this.idContinueTimes = idContinueTimes;

this.idDateOut = idDateOut;

this.idDateRetPlan = idDateRetPlan;

this.isHasReturn = isHasReturn;

}

### 6.3 Dao层类设计

每个数据库表对应一个Dao层类，主要实现该表的插删改查操作。

其中每个Dao层都有一个static块，来链接数据库

static Connection conn=null;

static{

DatabaseCon databaseCon=new DatabaseCon();

try {

conn=databaseCon.getCon();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

#### TB\_ReaderTypeDao类

1. **findAllType()函数用来查找数据库中所有的读者类型，实现代码如下：**

//查找全部类型

public List<TB\_ReaderType> findAllType() throws SQLException{

String sql="select \* from TB\_ReaderType";

PreparedStatement ps=conn.prepareStatement(sql);

List<TB\_ReaderType> typeList=new ArrayList<TB\_ReaderType>();

ResultSet rs=ps.executeQuery();

TB\_ReaderType type;

while(rs.next()){

int rdType=rs.getInt("rdType");//读者类别

String rdTypeName=rs.getString("rdTypeName");//类别名称

int canLendQty=rs.getInt("canLendQty");//可借书数量

int canLendDay=rs.getInt("canLendDay");//可借书天数

int canContinueTimes=rs.getInt("canContinueTimes");//可续借次数

float punishRate=rs.getFloat("punishRate");//罚款率

int dateValid=rs.getInt("dateValid");//证书有效期

type=new TB\_ReaderType(rdType, rdTypeName, canLendQty, canLendDay, canContinueTimes, punishRate, dateValid);

typeList.add(type);

}

return typeList;

}

1. findType()方法用来查找读者类型编号所对应的读者类型的所有数据：实现代码如下：

//查找对应类型

public TB\_ReaderType findType(int rdType) throws SQLException{

String sql="select \* from TB\_ReaderType where rdType=?";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setInt(1, rdType);

ResultSet rs=ps.executeQuery();

TB\_ReaderType type = null;

while(rs.next()){

// int rdType=rs.getInt("rdType");//读者类别

String rdTypeName=rs.getString("rdTypeName");//类别名称

int canLendQty=rs.getInt("canLendQty");//可借书数量

int canLendDay=rs.getInt("canLendDay");//可借书天数

int canContinueTimes=rs.getInt("canContinueTimes");//可续借次数

float punishRate=rs.getFloat("punishRate");//罚款率

int dateValid=rs.getInt("dateValid");//证书有效期

type=new TB\_ReaderType(rdType, rdTypeName, canLendQty, canLendDay, canContinueTimes, punishRate, dateValid);

}

return type;

}

1. updateType()用来更新读者类型包含的数据，实现代码如下：

//更新类型数据

public int updateType(TB\_ReaderType type) throws SQLException {

String sql="update TB\_ReaderType set rdTypeName=? , canLendQty=?,"

+ "canLendDay=?,canContinueTimes=?,punishRate=?,dateValid=? where rdType=?";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setString(1, type.getRdTypeName());

ps.setInt(2, type.getCanLendQty());

ps.setInt(3, type.getCanLendDay());

ps.setInt(4, type.getCanContinueTimes());

ps.setFloat(5, type.getPunishRate());

ps.setInt(6, type.getDateValid());

ps.setInt(7, type.getRdType());

return ps.executeUpdate();

}

}

#### TB\_ReaderDao类

1. **findReader()根据输入的列名与列对应的数据来查找学生数据，使用list来实现多列查询，当list为空时将查找所有学生。实现代码：**

//根据姓名等信息查找学生信息

public List<TB\_Reader> findReader(List<String> column,List<String> info) throws SQLException{

String sql1="select \* from TB\_Reader ";

String sql2="select \* from TB\_Reader where ";

String sql=sql1;

if(column.size()==0){

sql=sql1;

}else{

for(int i=0;i<column.size();i++){

if(i!=column.size()-1)

sql2=sql2+" "+column.get(i)+" "+"like "+" '%"+info.get(i)+"%' "+" and ";

if(i==column.size()-1)

sql2=sql2+" "+column.get(i)+" "+"like "+" '%"+info.get(i)+"%' ";

}

sql=sql2;

}

PreparedStatement ps=conn.prepareStatement(sql);

ResultSet rs=ps.executeQuery();

List<TB\_Reader> reader=new ArrayList<TB\_Reader >();

while(rs.next()){

int rdId=rs.getInt("rdId");//读者编号

String rdName=rs.getString("rdName");//姓名

String rdSex=rs.getString("rdSex");//性别

int rdType=rs.getInt("rdType");//读者类别

String rdDept=rs.getString("rdDept");//单位名称

String rdPhone=rs.getString("rdPhone");//电话

String rdEmail=rs.getString("rdEmail");//电子邮箱

Date rdDateReg=rs.getDate("rdDateReg");//读者登记日期

String rdPhoto=rs.getString("rdPhoto");//证件照片

String rdStatus=rs.getString("rdStatus");//证件状态

int rdBorrowQty=rs.getInt("rdBorrowQty");//已借书数量

String rdPwd=rs.getString("rdPwd");//读者密码

int rdAdminRoles=rs.getInt("rdAdminRoles");//管理角色

TB\_Reader read=new TB\_Reader(rdId, rdName, rdSex, rdType, rdDept, rdPhone, rdEmail,

rdDateReg, rdPhoto, rdStatus, rdBorrowQty, rdPwd, rdAdminRoles);

reader.add(read);

}

return reader;

}

1. InsertReader()方法是向数据库中添加新的读者信息，主要使用于新生入馆的功能，代码实现：

//新生入馆

public int insertReader(TB\_Reader reader) throws SQLException{

String sql="insert into TB\_Reader(rdId,rdName,rdSex,"

+ "rdType,rdDept,rdPhone,rdEmail,rdDateReg,"

+ "rdPhoto,rdStatus,rdAdminRoles) "

+ " values (?,?,?,?,?,?,?,?,?,?,?)";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setInt(1, reader.getRdId());

ps.setString(2, reader.getRdName());

ps.setString(3, reader.getRdSex());

ps.setInt(4, reader.getRdType());

ps.setString(5, reader.getRdDept());

ps.setString(6, reader.getRdPhone());

ps.setString(7, reader.getRdEmail());

ps.setDate(8, reader.getRdDateReg());

ps.setString(9, reader.getRdPhoto());

ps.setString(10, reader.getRdStatus());

// ps.setInt(11, reader.getRdBorrowQty());

// ps.setString(12, reader.getRdPwd());

ps.setInt(11, reader.getRdAdminRoles());

return ps.executeUpdate();

}

1. UpdateReader()用于更新已存在的读者信息，实现代码:

//更新读者信息

public int updateReader(TB\_Reader reader) throws SQLException{

String sql="update TB\_Reader set rdName=?,rdSex=?,rdType=?,rdDept=?,rdPhone=?,rdEmail=?,rdDateReg=?,rdPhoto=?,"

+ "rdStatus=?,rdBorrowQty=?,rdPwd=?,rdAdminRoles=? where rdId=?";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setString(1, reader.getRdName());

ps.setString(2, reader.getRdSex());

ps.setInt(3, reader.getRdType());

ps.setString(4, reader.getRdDept());

ps.setString(5, reader.getRdPhone());

ps.setString(6, reader.getRdEmail());

ps.setDate(7, reader.getRdDateReg());

ps.setString(8, reader.getRdPhoto());

ps.setString(9, reader.getRdStatus());

ps.setInt(10, reader.getRdBorrowQty());

ps.setString(11, reader.getRdPwd());

ps.setInt(12, reader.getRdAdminRoles());

ps.setInt(13, reader.getRdId());

return ps.executeUpdate();

}

#### TB\_BookDao类

1. findBook()**根据输入的列名与列对应的数据来查找图书数据，使用list来实现多列查询，当list为空时将查找所有图书。实现代码：**

//查找书籍

public List<TB\_Book> findBook(List<String> column,List<String> info) throws SQLException{

String sql1="select \* from TB\_Book ";

String sql2="select \* from TB\_Book where ";

String sql=sql1;

if(column.size()==0){

sql=sql1;

}else{

for(int i=0;i<column.size();i++){

if(i!=column.size()-1)

sql2=sql2+" "+column.get(i)+" "+"like "+" '%"+info.get(i)+"%' "+" and ";

if(i==column.size()-1)

sql2=sql2+" "+column.get(i)+" "+"like "+" '%"+info.get(i)+"%' ";

}

sql=sql2;

}

PreparedStatement ps=conn.prepareStatement(sql);

ResultSet rs=ps.executeQuery();

List<TB\_Book> book\_list=new ArrayList<TB\_Book>();

while(rs.next()){

int bkId=rs.getInt("bkId");

String bkCode=rs.getString("bkCode");//图书编号

String bkName=rs.getString("bkName");//书名

String bkAuthor=rs.getString("bkAuthor");//作者

String bkPress=rs.getString("bkPress");//出版社

Date bkDatePress=rs.getDate("bkDatePress");//出版日期

String bkISBN=rs.getString("bkISBN");//ISBN书号

String bkCatalog=rs.getString("bkCatalog");//分类号

int bkLanguage=rs.getInt("bkLanguage");//语言

int bkPages=rs.getInt("bkPages");//页数

BigDecimal bkPrice=rs.getBigDecimal("bkPrice");//价格

Date bkDateIn=rs.getDate("bkDateIn");//入馆日期

String bkBrief=rs.getString("bkBrief");//内容介绍

String bkCover=rs.getString("bkCover");//图书封面照片

String bkStatus=rs.getString("bkStatus");//图书状态

TB\_Book book=new TB\_Book(bkId,bkCode, bkName, bkAuthor, bkPress, bkDatePress, bkISBN, bkCatalog,

bkLanguage, bkPages, bkPrice, bkDateIn, bkBrief, bkCover, bkStatus);

book\_list.add(book);

}

return book\_list;

}

1. insertBook()方法是向数据库中添加新的图书信息，主要使用于新书入库的功能，代码实现：

//新书入库

public int insertBook(TB\_Book book) throws SQLException{

String sql="insert into TB\_Book(bkCode, bkName, bkAuthor, bkPress, bkDatePress, bkISBN,"

+ " bkCatalog, bkLanguage, bkPages, bkPrice, bkDateIn, bkBrief, bkCover, bkStatus) values (?,?,?,?,?,?,?,?,?,?,?,?,?,?)";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setString(1, book.getBkCode());

ps.setString(2, book.getBkName());

ps.setString(3, book.getBkAuthor());

ps.setString(4, book.getBkPress());

ps.setDate(5, book.getBkDatePress());

ps.setString(6, book.getBkISBN());

ps.setString(7,book.getBkCatalog());

ps.setInt(8, book.getBkLanguage());

ps.setInt(9, book.getBkPages());

ps.setBigDecimal(10, book.getBkPrice());

ps.setDate(11, book.getBkDateIn());

ps.setString(12, book.getBkBrief());

ps.setString(13,book.getBkCover());

ps.setString(14, book.getBkStatus());

return ps.executeUpdate();

}

1. 用于更新已存在的图书信息，实现代码:

//书籍维护

public int updateBook(TB\_Book book) throws SQLException{

String sql="update TB\_Book set bkName=?, bkAuthor=?, bkPress=?, bkDatePress=?, bkISBN=?,"

+ " bkCatalog=?, bkLanguage=?, bkPages=?, bkPrice=?, bkDateIn=?, bkBrief=?, bkCover=? where bkCode=? ";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setString(1, book.getBkName());

ps.setString(2, book.getBkAuthor());

ps.setString(3, book.getBkPress());

ps.setDate(4, book.getBkDatePress());

ps.setString(5, book.getBkISBN());

ps.setString(6,book.getBkCatalog());

ps.setInt(7, book.getBkLanguage());

ps.setInt(8, book.getBkPages());

ps.setBigDecimal(9, book.getBkPrice());

ps.setDate(10, book.getBkDateIn());

ps.setString(11, book.getBkBrief());

ps.setString(12,book.getBkCover());

ps.setString(13, book.getBkCode());

return ps.executeUpdate();

}

1. ChangBook()用于实现借书还书的功能,实现代码如下：

//借还书籍

public int changBook(TB\_Book book) throws SQLException{

String sql="update TB\_Book set bkStatus=? where bkId=? ";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setString(1, book.getBkStatus());

ps.setInt(2, book.getBkId());

return ps.executeUpdate();

}

#### TB\_BorrowDao类

1. FindAllBorrow()用于查找当前读者所有已借未还书籍，实现代码如下：

//根据rdid查找已借未还书籍

public List<TB\_Borrow> findAllBorrow(int rdId) throws SQLException{

String sql="select \* from TB\_Borrow where rdId=?";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setInt(1, rdId);

ResultSet rs=ps.executeQuery();

List<TB\_Borrow> borrowList=new ArrayList<TB\_Borrow>();

TB\_Borrow borrow=null;

while(rs.next()){

if(!rs.getBoolean("isHasReturn")){

int bkID=rs.getInt("bkID");//图书序号

int idContinueTimes=rs.getInt("idContinueTimes");//续借次数

Date idDateOut=rs.getDate("idDateOut");//借书日期

Date idDateRetPlan=rs.getDate("idDateRetPlan");//应还日期

boolean isHasReturn=rs.getBoolean("isHasReturn");//是否还书

borrow=new TB\_Borrow(rdId, bkID, idContinueTimes, idDateOut, idDateRetPlan, isHasReturn);

borrowList.add(borrow);

}

}

return borrowList;

}

1. FindBookBorrow()根据书籍的编号来查找图书，实现代码:

//根据bkId查找书籍

public TB\_Borrow findBookBorrow(int bkId) throws SQLException{

String sql="select \* from TB\_Borrow where bkId=?";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setInt(1,bkId);

ResultSet rs=ps.executeQuery();

TB\_Borrow borrow=null;

while(rs.next()){

if(!rs.getBoolean("isHasReturn")){

int rdId=rs.getInt("rdId");//读者序号

int idContinueTimes=rs.getInt("idContinueTimes");//续借次数

Date idDateOut=rs.getDate("idDateOut");//借书日期

Date idDateRetPlan=rs.getDate("idDateRetPlan");//应还日期

boolean isHasReturn=rs.getBoolean("isHasReturn");//是否还书

borrow=new TB\_Borrow(rdId, bkId, idContinueTimes, idDateOut, idDateRetPlan, isHasReturn);

}

}

return borrow;

}

1. InsertBorrow()是用于读者借书时，将借书信息插入数据库，实现代码：

//插入数据，借书

public int insertBorrow(TB\_Borrow borrow) throws SQLException{

String sql="insert into TB\_Borrow(rdID,bkID, idContinueTimes, idDateOut, idDateRetPlan,isHasReturn,operatorLend) "

+ "values(?,?,?,?,?,?,?) ";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setInt(1, borrow.getRdID());

ps.setInt(2,borrow.getBkID());

ps.setInt(3, borrow.getIdContinueTimes());

ps.setDate(4, borrow.getIdDateOut());//借书日期

ps.setDate(5, borrow.getIdDateRetPlan());//应还日期

ps.setBoolean(6, borrow.isHasReturn());

ps.setString(7, borrow.getOperatorLend());

return ps.executeUpdate();

}

1. ChangBorrow()是更改借书数据表中图书是否归还，并更新其超期时间及罚款金额，实现代码：

//更新书籍是否归还

public int changBorrow(TB\_Borrow borrow) throws SQLException{

String sql="update TB\_Borrow set isHasReturn=?,idDateRetAct=?,idOverDay=?,idOverMoney=?,operatorRet=? where bkId=?";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setBoolean(1, borrow.isHasReturn());

ps.setDate(2, borrow.getIdDateRetAct());

ps.setInt(3, borrow.getIdOverDay());

ps.setBigDecimal(4, borrow.getIdOverMoney());

ps.setString(5, borrow.getOperatorLend());//还书操作

ps.setInt(6, borrow.getBkID());

return ps.executeUpdate();

}

1. NextBorrow()实现续借功能，就是更新一下应还时间，以及已借次数，实现代码：

//续借

public int nextBorrow(TB\_Borrow borrow) throws SQLException{

String sql="update TB\_Borrow set idDateRetPlan=?,operatorLend=?,idContinueTimes=? where bkId=?";

PreparedStatement ps=conn.prepareStatement(sql);

ps.setDate(1, borrow.getIdDateRetPlan());

ps.setString(2, borrow.getOperatorLend());

ps.setInt(3, borrow.getIdContinueTimes()+1);

ps.setInt(4, borrow.getBkID());

return ps.executeUpdate();

}

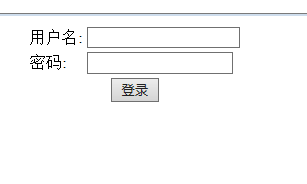
### 6.4view实现以及功能实现

除主窗体要考虑整体功能结构外，其它功能都是按用例逐一进行设计和实现的。部分用例的事件流相似且相关，可集中在同一个窗体内实现，如：借书证挂失、解除、补办与注销。

#### 用户登录

登录方法的实现

账号密码与数据库中的对比,相同，则跳转至主页面，否则登录失败



登录的数据通过表单提交给LoginServlet表单代码如下：

<form action="LoginServlet" method="post" >

<table align="center">

<tr>

<td>用户名:</td>

<td><input type="text" name="username" /></td>

</tr>

<tr>

<td>密码:</td>

<td><input type="password" name="pwd" /></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="Submit" value="登录" />

</td>

</tr>

<tr>

<c:if test="${adminType!=null }">

<c:out value="${adminType }"></c:out>

<c:remove var="adminType" scope="session" />

</c:if>

</tr>

</table>

</form>

当按下登录按钮时，页面将表单中的username与pwd的数据提交LoginServlet，让LoginServlet链接数据库对比数据来判断是否登录成功，登录成功则跳转至主页面，并设置了多个session数据，让后序的页面及servlet来判断登录者的身份，其中admin表示登录者的编号，adminType表示登录者的权限；失败则显示登录失败，设置session来提示登录者，登录失败，功能实现代码：

try {

HttpSession session=request.getSession();

String str=request.getParameter("username");

int rdId=Integer.parseInt(str);

String rdPwd=request.getParameter("pwd");

TB\_ReaderDao readerDao=new TB\_ReaderDao();

TB\_Reader reader=readerDao.findReaderById(rdId);

TB\_ReaderTypeDao typeDao=new TB\_ReaderTypeDao();

List<TB\_ReaderType> typeList= typeDao.findAllType();

if(reader.getRdPwd().equals(rdPwd)){

session.setAttribute("admin", rdId);

session.setAttribute("adminTypeList", typeList);//类型列表

session.setAttribute("adminType", reader.getRdAdminRoles());

response.sendRedirect("main.jsp");

}else{

session.setAttribute("adminType","登录失败!");

response.sendRedirect("login.jsp");

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

显示登录失败页面代码如下：

<c:if test="${adminType!=null }">

<c:out value="${adminType }"></c:out>

<c:remove var="adminType" scope="session" />

</c:if>

#### Main主窗体界面

登录成功后显示的界面，根据用户类型不同，显示的操作不同,点击操作按钮，会跳转至不同的操作页面，下面显示的分别是系统管理员、读者管理员、读者、图书管理员与借阅管理员对应的界面









当用户登录后，页面中会根据读者的权限编号显示不同的按钮，每个按钮对应不同功能，筛选管理员功能通过jstl的c:if表达式实现，代码如下：

<tr>

<td><input type="button" value="书籍查找" onclick="location='findBook.jsp'"/></td>

<td><input type="button" value="密码更改" onclick="location='myPassword.jsp'"/></td>

<c:if test="${adminType==1||adminType==0 }">

<td><input type="button"id="reader"value="读者查找" onclick="location='findReader.jsp'"/></td>

<td><input type="button" id="reader" value="新生入馆" onclick="location='insertReader.jsp'"/></td>

<td><input type="button" id="reader"value="信息更改" onclick="location='findReader.jsp'"/></td>

<td><input type="button" id="reader"value="类别管理" onclick="location='FindReaderTypeServlet'"/></td>

<td><input type="button" value="重置密码" onclick="location='password.jsp'"/></td>

</c:if>

<c:if test="${adminType==2||adminType==0 }">

<td><input type="button" id="book" value="新书入库" onclick="location='insertBook.jsp'"/></td>

<td><input type="button" id="book" value="图书维护" onclick="location='findBook.jsp'"/></td>

</c:if>

<c:if test="${adminType==3||adminType==0 }">

<td><input type="button" id="borrow"value="借阅管理" onclick="location='borrowBook.jsp'"/></td>

</c:if>

<c:if test="${adminType==0 }">

<td><input type="button" value="权限管理" onclick="location='adminRoles.jsp'"/></td>

</c:if>

<td><input type="button" value="退出" onclick="location='ExitServlet'"/></td>

</tr>

#### 书籍查找



分为简单查找和高级查找，简单查找根据书名或作者名，高级查找根据书名、作者、出版社、分类号、出版日期等查找，当用户有图书管理权限时会出现维护书籍的按钮（此功能通过c:if实现），当输入信息后，按下搜索按钮，会提交信息到FindBookServlet,servlet会分辨是简单还是高级查询，然后输出图书信息，servlet中查找方法代码如下：

try {

HttpSession session=request.getSession();

List<String> column=new ArrayList<String>();

List<String> info=new ArrayList<String>();

List<TB\_Book> bookList=new ArrayList<TB\_Book>();

request.setCharacterEncoding("utf-8");

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

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

if(findType!=null&&findInfo!=null){

column.add(findType);

info.add( findInfo);

}else{

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

if(!bkName.equals("")){

column.add("bkName");

info.add(bkName);

}

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

if(!(bkAuthor.equals(""))){

column.add("bkAuthor");

info.add(bkAuthor);

}

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

if(!bkName.equals("")){

column.add("bkPress");

info.add(bkPress);

}

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

if(!bkCatalog.equals("")){

column.add("bkCatalog");

info.add(bkCatalog);

}

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

if(!bkDatePress.equals("")){

column.add("bkDatePress");

info.add(bkDatePress);

}

}

TB\_BookDao bookDao=new TB\_BookDao();

bookList=bookDao.findBook(column, info);

session.setAttribute("findBookList", bookList);

response.sendRedirect("findBook.jsp");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

页面采用表单提交，使用jstl表达式输出，页面代码如下：

</form>

<h3>简单查询</h3>

<form action="FindBookServlet" method="post">

<table>

<tr>

<td>

<select name="findType">

<option value="bkName">书名</option>

<option value="bkAuthor">作者</option>

</select>

<input type="text" name="findInfo"/>

</td>

<td><input type="submit" value="搜索"/></td>

</tr>

</table>

</form>

<h3>高级查询</h3>

<form action="FindBookServlet" method="post">

<table>

<tr>

<td>图书名称</td>

<td><input type="text" name="bkName"/></td>

<td>图书作者</td>

<td><input type="text" name="bkAuthor"/></td>

<td>出版社名</td>

<td><input type="text" name="bkPress"/></td>

</tr>

<tr>

<td>分类号</td>

<td><input type="text" name="bkCatalog"/></td>

<td>出版日期</td>

<td><input type="text" name="bkDatePress" /></td>

<td><input type="submit" value="搜索"/></td>

</tr>

</table>

</form>

<table>

<tr>

<td>ID</td>

<td>索书号</td>

<td>书名</td>

<td>作者</td>

<td>出版社</td>

<td>出版日期</td>

<td>ISBN</td>

<td>分类号</td>

<td>语种</td>

<td>页数</td>

<td>价格</td>

<td>入馆时间</td>

<td>状态</td>

</tr>

<c:forEach items="${findBookList}" var="items">

<tr>

<td>${items.getBkId()}</td>

<td>${items.getBkCode()}</td>

<td>${items.getBkName() }</td>

<td>${items.getBkAuthor() }</td>

<td>${items.getBkPress() }</td>

<td>${items. getBkDatePress() }</td>

<td>${items.getBkISBN() }</td>

<td>${items.getBkCatalog() }</td>

<td>${items.getBkLanguage()}</td>

<td>${items.getBkPages() }</td>

<td>${items.getBkPrice() }</td>

<td>${items.getBkDateIn() }</td>

<td>${items.getBkStatus() }</td>

<c:if test="${adminType!=4 }">

<td><a href="ShowUpdateBookServlet?bkCode=${items.getBkCode()}"><input type="button" value="维护书籍"/></a></td>

</c:if>

</tr>

</c:forEach>

</table>

#### 新书入馆



将新增图书信息通过后台添加至数据库，输入信息后，点击提交，后台获取信息，输入数据库中，其中后台代码为

try {

HttpSession session=request.getSession();

request.setCharacterEncoding("utf-8");

String bkCode=request.getParameter("bkCode");//图书编号

String bkName=request.getParameter("bkName");//书名

String bkAuthor=request.getParameter("bkAuthor");//作者

String bkPress=request.getParameter("bkPress");//出版社

String getBkDatePress=request.getParameter("bkDatePress");

Date bkDatePress= Date.valueOf(getBkDatePress);//出版日期

String bkISBN=request.getParameter("bkISBN");//ISBN书号

String bkCatalog=request.getParameter("bkCatalog");//分类号

String getBkLanguage=request.getParameter("bkLanguage");

int bkLanguage=Integer.parseInt(getBkLanguage);//语言

String getBkPages=request.getParameter("bkPages");

int bkPages=Integer.parseInt(getBkPages);//页数

String getBkPrice=request.getParameter("bkPrice");

BigDecimal bkPrice=new BigDecimal(getBkPrice);//价格

Date bkDateIn=new Date(System.currentTimeMillis());

// Date bkDateIn=(Date) format.parse(format.format(new java.util.Date()));//入馆日期

String bkBrief=request.getParameter("bkBrief");//内容介绍

String getBkCover=request.getParameter("bkCover");//图书封面照片

String bkCover=bkName+".jpg";

String mybkCover= getServletContext().getRealPath("/")+"/libraryImage/book/"+bkName+".jpg";

copyImge(getBkCover, mybkCover);//图书封面照片另存

String bkStatus="在馆";//图书状态

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

int Count=Integer.parseInt(bkCount);

TB\_Book book=new TB\_Book(bkCode, bkName, bkAuthor, bkPress, bkDatePress, bkISBN, bkCatalog, bkLanguage, bkPages, bkPrice, bkDateIn, bkBrief, bkCover, bkStatus);

TB\_BookDao bookDao=new TB\_BookDao();

for(int i=0;i<Count;i++){

if(bookDao.insertBook(book)!=0){

session.setAttribute("insertError", "插入成功");

}else{

session.setAttribute("insertError", "插入失败");

}

}

response.sendRedirect("insertBook.jsp");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

界面实现代码为：

<h4 align="center">新书入库</h4>

<form action="InsertBookServlet" method="post">

<c:if test="${insertError!=null }">

<h3><c:out value="${insertError }"></c:out></h3>

<c:remove var="insertError" scope="session"/>

</c:if>

<table width="1000px" align="left">

<tr>

<td width="300px" style="float:left;margin: 0px;padding: 0px;">

<table>

<!-- <tr>

<td>图书起始序号</td>

<td><input type="text" /></td>

</tr>-->

<tr>

<td > 图 书名 称</td>

<td><input type="text" name="bkName"/></td>

</tr>

<tr>

<td>索 书 号</td>

<td><input type="text" name="bkCode"/></td>

</tr>

<tr>

<td>作者</td>

<td><input type="text" name="bkAuthor"/></td>

<tr>

<td>出版 社</td>

<td><input type="text" name="bkPress"/></td>

</tr>

<tr>

<td>出 版日期 </td>

<td><input type="text" name="bkDatePress" value="1900-01-01" style="color:#CDC9C9;"onfocus="if (value =='1900-01-01'){this.style.color='#000';value =''}"

onblur="if (value ==''){this.style.color='#CDC9C9';value='1900-01-01'}" /></td>

</tr>

<tr>

<td>ISBN</td>

<td><input type="text" name="bkISBN"/></td>

</tr>

<tr>

<td>分类号</td>

<td><input type="text" name="bkCatalog"/></td>

</tr>

<tr>

<td>语种 </td>

<td>

<select name="bkLanguage">

<option value="0">中文</option>

<option value="1">英文</option>

<option value="2">日文</option>

<option value="3">俄文</option>

<option value="4">德文</option>

<option value="5">法文</option>

</select>

</td>

</tr>

<tr>

<td>图 书页面</td>

<td><input type="text" name="bkPages" /></td>

</tr>

<tr>

<td>图书 价格 </td>

<td><input type="text" name="bkPrice"/></td>

</tr>

<tr>

<td>图 书本书：</td>

<td><input type="text" name="bkCount"/></td>

</tr>

</table>

</td>

<td width="300px" style="float:left;margin: 0px;padding: 0px;">

<label>图书简介</label>

<textarea rows="20" cols="30" name="bkBrief"></textarea><br />

</td>

<td width="300px" style="float:right;margin: 0px;padding: 0px;">

<div id="main">

<div class="demo">

<label>请选择一个图像文件：</label>

<input type="file" id="file\_input" name="bkCover" />

<br />

<div id="result">

<!-- 这里用来显示读取结果 -->

</div>

</div>

</div>

<input type="submit" value="提交" />

</td>

</tr>

</table>

</form>

其中显示本地图片使用了javascript,代码如下：

<script type="text/javascript">

var result = document.getElementById("result");

var input = document.getElementById("file\_input");

if(typeof FileReader === 'undefined'){

result.innerHTML = "抱歉，你的浏览器不支持 FileReader";

input.setAttribute('disabled','disabled');

}else{

input.addEventListener('change',readFile,false);

}

function readFile(){

var file = this.files[0];

if(!/image\/\w+/.test(file.type)){

alert("请确保文件为图像类型");

return false;

}

var reader = new FileReader();

reader.readAsDataURL(file);

reader.onload = function(e){

result.innerHTML = '<img src="'+this.result+'" alt="" width="240px" height="300px"/>'

}

}

</script>

而在后台中获取图片地址后，使用copyImge方法将本地图片存在服务器端的相对地址上，具体方法实现为：

private static FileOutputStream fout= null;

private static FileInputStream fin = null;

public static void copyImge(String inName,String outName) {

try {

fout = new FileOutputStream(outName);//复制文件

fin = new FileInputStream(inName);//源文件

byte[] buf = new byte[1024];//缓冲区

int len = 0;

while ((len = fin.read(buf)) != -1) {

fout.write(buf, 0, len);//复制

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (fin != null)

fin.close();

} catch (IOException e) {

e.printStackTrace();

}

try {

if (fout != null)

fout.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

#### 维护书籍

更新页面数据



维护书籍的页面实现方法与新书入库相差不大，只是将查找页面中后台获取的图书数据打印出来，更新数据的方法就是使用update来更新数据库数据，这里使用了c:foreach帮助循环打印，实现代码如下：

<form action="UpdateBookServlet" method="post">

<table width="1000px" align="left">

<tr>

<td width="300px" style="float:left;margin: 0px;padding: 0px;">

<table>

<!-- <tr>

<td>图书起始序号</td>

<td><input type="text" /></td>

</tr>-->

<tr>

<td > 图 书名 称</td>

<td><input type="text" name="bkName" value="${updateBook.getBkName() }"/></td>

</tr>

<tr>

<td>索 书 号</td>

<td><input type="text" name="bkCode" value="${updateBook.getBkCode() }"/></td>

</tr>

<tr>

<td>作者</td>

<td><input type="text" name="bkAuthor" value="${updateBook.getBkAuthor() }"/></td>

<tr>

<td>出版 社</td>

<td><input type="text" name="bkPress" value="${updateBook.getBkPress() }"/></td>

</tr>

<tr>

<td>出 版日期 </td>

<td><input type="text" name="bkDatePress" value="${updateBook.getBkDatePress() }" /></td>

</tr>

<tr>

<td>ISBN</td>

<td><input type="text" name="bkISBN" value="${updateBook.getBkISBN() }"/></td>

</tr>

<tr>

<td>分类号</td>

<td><input type="text" name="bkCatalog" value="${updateBook.getBkCatalog() }"/></td>

</tr>

<tr>

<td>语种 </td>

<td>

<select name="bkLanguage" id="bkLanguage">

<option value="0">中文</option>

<option value="1">英文</option>

<option value="2">日文</option>

<option value="3">俄文</option>

<option value="4">德文</option>

<option value="5">法文</option>

</select>

</td>

</tr>

<tr>

<td>图 书页面</td>

<td><input type="text" name="bkPages" value="${updateBook.getBkPages() }"/></td>

</tr>

<tr>

<td>图书 价格 </td>

<td><input type="text" name="bkPrice" value="${updateBook.getBkPrice() }"/></td>

</tr>

</table>

</td>

<td width="300px" style="float:left;margin: 0px;padding: 0px;">

<label>图书简介</label>

<textarea rows="20" cols="30" name="bkBrief" >${updateBook.getBkBrief()}</textarea><br />

</td>

<td width="300px" style="float:right;margin: 0px;padding: 0px;">

<div id="main">

<div class="demo">

<label>请选择一个图像文件：</label>

<input type="file" id="file\_input" name="bkCover" />

<br />

<div id="result">

<!-- 这里用来显示读取结果 -->

<img src="${updateBook.getBkCover()} " width="240px" height="300px"/>

</div>

</div>

</div>

<input type="submit" value="更新数据" />

</td>

</tr>

</table>

</form>

后台更新数据的代码与新书入库的后台代码相差不大，只是将插入改为更新，实现代码如下：

request.setCharacterEncoding("utf-8");

String bkCode=request.getParameter("bkCode");//图书编号

String bkName=request.getParameter("bkName");//书名

String bkAuthor=request.getParameter("bkAuthor");//作者

String bkPress=request.getParameter("bkPress");//出版社

String getBkDatePress=request.getParameter("bkDatePress");

// SimpleDateFormat format=new SimpleDateFormat("yyyy-MM");

Date bkDatePress= Date.valueOf(getBkDatePress);//出版日期

String bkISBN=request.getParameter("bkISBN");//ISBN书号

String bkCatalog=request.getParameter("bkCatalog");//分类号

String getBkLanguage=request.getParameter("bkLanguage");

int bkLanguage=Integer.parseInt(getBkLanguage);//语言

String getBkPages=request.getParameter("bkPages");

int bkPages=Integer.parseInt(getBkPages);//页数

String getBkPrice=request.getParameter("bkPrice");

BigDecimal bkPrice=new BigDecimal(getBkPrice);//价格

Date bkDateIn=new Date(System.currentTimeMillis());

// Date bkDateIn=(Date) format.parse(format.format(new java.util.Date()));//入馆日期

String bkBrief=request.getParameter("bkBrief");//内容介绍

String getBkCover=request.getParameter("bkCover");//图书封面照片

String bkCover=bkName+".jpg";

String mybkCover= getServletContext().getRealPath("/")+"/libraryImage/book/"+bkName+".jpg";

copyImge(getBkCover, mybkCover);//图书封面照片另存

String bkStatus="在馆";//图书状态

TB\_Book book=new TB\_Book(bkCode, bkName, bkAuthor, bkPress, bkDatePress, bkISBN, bkCatalog, bkLanguage, bkPages, bkPrice, bkDateIn, bkBrief, bkCover, bkStatus);

TB\_BookDao bookDao=new TB\_BookDao();

bookDao.updateBook(book);

response.sendRedirect("findBook.jsp");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

#### 读者查找



输入读者类别、单位以及姓名查找,读者管理员可以从此页面进入读者信息更改

搜索功能后台实现：

try {

HttpSession session=request.getSession();

List<String> column=new ArrayList<String>();

List<String> info=new ArrayList<String>();

List<TB\_Reader> readerList=new ArrayList<TB\_Reader>();

request.setCharacterEncoding("utf-8");

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

if(!rdType.equals("")){

column.add("rdType");

info.add(rdType);

}

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

if(!rdDept.equals("")){

column.add("rdDept");

info.add(rdDept);

}

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

if(!rdName.equals("")){

column.add("rdName");

info.add(rdName);

}

TB\_ReaderDao readerDao=new TB\_ReaderDao();

readerList=readerDao.findReader(column, info);

session.setAttribute("readerList",readerList);

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

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

前端输出实现：

</form>

<h4 align="center">读者查找</h4>

<div class="main" align="center">

<form action="FindReaderServlet" method="post">

<table>

<tr>

<td>读者类别：<select name="rdType">

<c:forEach items="${adminTypeList }" var="items">

<option value="${items.getRdType() }">${items.getRdTypeName()}</option>

</c:forEach>

</select>

</td>

<td>单位：<select name="rdDept">

<option>计算机学院</option>

<option>管理学院</option>

<option>经济学院</option>

</select>

</td>

<td>姓名：<input type="text" name="rdName"></td>

<td><input type="submit" value="查询"></td>

<!-- <td> <input type="submit" value="EXCL"></td>-->

</tr>

</table>

</form>

<table width="700px" border="1px">

<tr>

<td>ID</td>

<td>姓名</td>

<td>性别</td>

<td>类别</td>

<td>院系</td>

<td>电话</td>

<td>e-mail</td>

<td>状态</td>

<td>已借书</td>

<td>注册日期</td>

</tr>

<c:forEach items="${readerList}" var="items">

<tr>

<td>${items.getRdId()}</td>

<td>${items.getRdName()}</td>

<td>${items.getRdSex() }</td>

<td>${items.getRdType() }</td>

<td>${items.getRdDept() }</td>

<td>${items.getRdPhone()}</td>

<td>${items.getRdEmail() }</td>

<td>${items.getRdStatus()}</td>

<td>${items.getRdBorrowQty()}</td>

<td>${items.getRdDateReg()}</td>

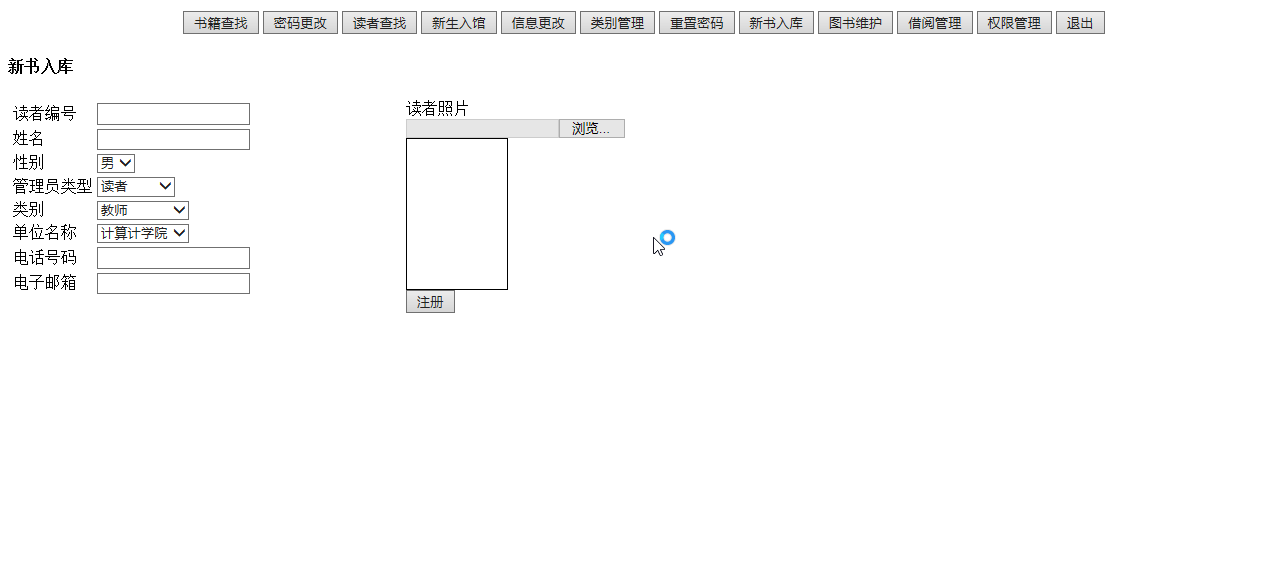
<td><a href="ShowUpdateReaderServlet?rdId=${items.getRdId()}"><input type="button" value="更新数据"/></a></td>

</tr>

</c:forEach>

</table>

#### 新生入馆



与新书入库相似，将新读者的信息经由后台输入数据库

页面实现：

<h4 align="left">新生入馆</h4>

<form action="InsertReaderServlet" method="post">

<c:if test="${insertError!=null }">

<h3><c:out value="${insertError }"></c:out></h3>

<c:remove var="insertError" scope="session"/>

</c:if>

<table width="700px" align="left">

<tr>

<td width="300px" style="float:left;margin: 0px;padding: 0px;">

<table>

<tr>

<td >读者编号</td>

<td><input type="text" name="rdId"/></td>

</tr>

<tr>

<td>姓名</td>

<td><input type="text" name="rdName"/></td>

</tr>

<tr>

<td>性别</td>

<td><select name="rdSex">

<option value="男">男</option>

<option value="女">女</option>

</select></td>

</tr>

<tr>

<td>管理员类型</td>

<td><select name="rdAdminRoles" id="rdAdminRoles">

<option value="0">系统管理</option>

<option value="1">读者管理</option>

<option value="2">图书管理</option>

<option value="3">借书管理</option>

<option value="4" selected="selected">读者</option>

</select>

</td>

</tr>

<tr>

<td>类别</td>

<td><select name="rdType">

<option value="10">教师</option>

<option value="20">本科生</option>

<option value="21">专科生</option>

<option value="30">硕士研究生</option>

<option value="31">博士研究生</option>

</select></td>

</tr>

<tr>

<td>单位名称</td>

<td><select name=*"rdDept"* id=*"rdDept"*>

<option value=*"计算计学院"* selected=*"selected"*>计算计学院</option>

<option value=*"管理学院"*>管理学院</option>

<option value=*"经济学院"*>经济学院</option>

</select>

</td>

</tr>

<tr>

<td>电话号码</td>

<td><input type="text" name="rdPhone"/></td>

</tr>

<tr>

<td>电子邮箱</td>

<td><input type="text" name="rdEmail"/></td>

</tr>

<tr>

</table>

</td>

<td width="300px" style="float:right;margin: 0px;padding: 0px;">

<div id="main">

<div class="demo">

<label>读者照片</label> <br/>

<input type="file" id="file\_input" name="rdPhoto" />

<br />

<div id="result">

<!-- 这里用来显示读取结果 -->

</div>

</div>

</div>

<input type="submit" value="注册" />

</td>

</tr>

</table>

</form>

获取本地照片js实现：

<script type="text/javascript">

var result = document.getElementById("result");

var input = document.getElementById("file\_input");

if(typeof FileReader === 'undefined'){

result.innerHTML = "抱歉，你的浏览器不支持 FileReader";

input.setAttribute('disabled','disabled');

}else{

input.addEventListener('change',readFile,false);

}

function readFile(){

var file = this.files[0];

if(!/image\/\w+/.test(file.type)){

alert("请确保文件为图像类型");

return false;

}

var reader = new FileReader();

reader.readAsDataURL(file);

reader.onload = function(e){

result.innerHTML = '<img src="'+this.result+'" alt="" width="100px" height="150px"/>'

}

}

</script>

后台功能实现：

try {

HttpSession session=request.getSession();

request.setCharacterEncoding("utf-8");

String getRdId=request.getParameter("rdId");

int rdId=Integer.parseInt(getRdId);//读者编号

String rdName=request.getParameter("rdName");//姓名

String rdSex=request.getParameter("rdSex");//性别

String getRdType=request.getParameter("rdType");

int rdType=Integer.parseInt(getRdType);//读者类别

String rdDept=request.getParameter("rdDept");//单位名称

String rdPhone=request.getParameter("rdPhone");//电话

String rdEmail=request.getParameter("rdEmail");//电子邮箱

Date rdDateReg=new Date(System.currentTimeMillis());//读者登记日期

String rdStatus="有效";//证件状态

// int rdBorrowQty=request.getParameter("rdBorrowQty");//已借书数量

// String rdPwd=request.getParameter("rdPwd");//读者密码

// int rdAdminRoles=request.getParameter("rdAdminRoles");//管理角色

String getRdAdminRoles=request.getParameter("rdAdminRoles");

int rdAdminRoles=Integer.parseInt(getRdAdminRoles);

String getRdPhoto=request.getParameter("rdPhoto");//证件照片

String rdPhoto=rdId+".jpg";

String myRdPhoto= getServletContext().getRealPath("/")+"/libraryImage/reader/"+rdId+".jpg";

copyImge(getRdPhoto, myRdPhoto);//图书封面照片另存

int rdBorrowQty=0;//已借书数量

String rdPwd="123";//读者密码

TB\_Reader reader=new TB\_Reader(rdId, rdName, rdSex, rdType, rdDept, rdPhone, rdEmail, rdDateReg, rdPhoto, rdStatus, rdBorrowQty, rdPwd, rdAdminRoles);

TB\_ReaderDao readerDao=new TB\_ReaderDao();

if(readerDao.insertReader(reader)!=0){

session.setAttribute("insertError", "插入成功");

}else{

session.setAttribute("insertError", "插入失败");

}

response.sendRedirect("insertReader.jsp");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

本地照片转存服务器端，实现代码：  
private static FileOutputStream fout= null;

private static FileInputStream fin = null;

public static void copyImge(String inName,String outName) {

try {

fout = new FileOutputStream(outName);//复制文件

fin = new FileInputStream(inName);//源文件

byte[] buf = new byte[1024];//缓冲区

int len = 0;

while ((len = fin.read(buf)) != -1) {

fout.write(buf, 0, len);//复制

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (fin != null)

fin.close();

} catch (IOException e) {

e.printStackTrace();

}

try {

if (fout != null)

fout.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

#### 信息更改



更新读者信息，与图书维护的实现相似，页面将信息打印出来，更改需要更改的信息，提交后台，后台更新数据库

页面实现：

<h4 align="left">信息更改</h4>

<form action="UpdateReaderServlet" method="post">

<table width="700px" align="left">

<tr>

<td width="300px" style="float:left;margin: 0px;padding: 0px;">

<table>

<tr>

<td >读者编号</td>

<td><input type="text" name="rdId" value="${updateReader.getRdId() }" readonly="readonly"/></td>

</tr>

<tr>

<td>姓名</td>

<td><input type="text" name="rdName" value="${updateReader.getRdName() }"/></td>

</tr>

<tr>

<td>性别</td>

<td><select name="rdSex" id="rdSex">

<option value="男">男</option>

<option value="女">女</option>

</select></td>

<tr>

<td>类别</td>

<td><select name="rdType" id="rdType">

<c:forEach items="${adminTypeList }" var="items">

<option value="${items.getRdType() }">${items.getRdTypeName()}</option>

</c:forEach>

</select></td>

</tr>

<tr>

<td>证件状态</td>

<td>

<select name="rdStatus" id="rdStatus">

<option value="注销">注销</option>

<option value="挂失">挂失</option>

<option value="有效">有效</option>

</select>

</td>

</tr>

<tr>

<td>单位名称</td>

<td><select name=*"rdDept"* id=*"rdDept"*>

<option value=*"计算计学院"* selected=*"selected"*>计算计学院</option>

<option value=*"管理学院"*>管理学院</option>

<option value=*"经济学院"*>经济学院</option>

</select>

</td>

</tr>

<tr>

<td>电话号码</td>

<td><input type="text" name="rdPhone" value="${updateReader.getRdPhone() }"/></td>

</tr>

<tr>

<td>电子邮箱</td>

<td><input type="text" name="rdEmail" value="${updateReader.getRdEmail() }"/></td>

</tr>

<tr>

</table>

</td>

<td width="300px" style="float:right;margin: 0px;padding: 0px;">

<div id="main">

<div class="demo">

<label>读者照片</label> <br/>

<input type="file" id="file\_input" name="rdPhoto" value="${updateReader.getRdPhoto() }" />

<br />

<div id="result">

<!-- 这里用来显示读取结果 -->

<img src="${updateReader.getRdPhoto() }" alt="" width="100px" height="150px"/>

</div>

</div>

</div>

<input type="submit" value="更改信息" />

</td>

</tr>

</table>

</form>

<script type="text/javascript">

var result = document.getElementById("result");

var input = document.getElementById("file\_input");

if(typeof FileReader === 'undefined'){

result.innerHTML = "抱歉，你的浏览器不支持 FileReader";

input.setAttribute('disabled','disabled');

}else{

input.addEventListener('change',readFile,false);

}

function readFile(){

var file = this.files[0];

if(!/image\/\w+/.test(file.type)){

alert("请确保文件为图像类型");

return false;

}

var reader = new FileReader();

reader.readAsDataURL(file);

reader.onload = function(e){

result.innerHTML = '<img src="'+this.result+'" alt="" width="100px" height="150px"/>'

}

}

for(var i=0;i<document.getElementById('rdType').options.length;i++){

if(document.getElementById('rdType').options[i].value=='${updateReader.getRdType()}'){

document.getElementById('rdType').options[i].selected=true;

}

}

for(var i=0;i<document.getElementById('rdSex').options.length;i++){

if(document.getElementById('rdSex').options[i].value=='${updateReader.getRdSex()}'){

document.getElementById('rdSex').options[i].selected=true;

}

}

for(var i=0;i<document.getElementById('rdStatus').options.length;i++){

if(document.getElementById('rdStatus').options[i].value=='${updateReader.getRdStatus()}'){

document.getElementById('rdStatus').options[i].selected=true;

}

}

**for**(**var** i=0;i<document.getElementById('rdDept').options.length;i++){

**if**(document.getElementById('rdDept').options[i].value=='${updateReader.getRdDept() }'){

document.getElementById('rdDept').options[i].selected=**true**;

}

}

</script>

页面功能实现：

try {

request.setCharacterEncoding("utf-8");

String getRdId=request.getParameter("rdId");

int rdId=Integer.parseInt(getRdId);//读者编号

TB\_ReaderDao readerDao=new TB\_ReaderDao();

readerDao.findReaderById(rdId);

String rdName=request.getParameter("rdName");//姓名

String rdSex=request.getParameter("rdSex");//性别

String getRdType=request.getParameter("rdType");

int rdType=Integer.parseInt(getRdType);//读者类别

String rdDept=request.getParameter("rdDept");//单位名称

String rdPhone=request.getParameter("rdPhone");//电话

String rdEmail=request.getParameter("rdEmail");//电子邮箱

Date rdDateReg=new Date(System.currentTimeMillis());//读者登记日期

String rdStatus=null;

if(readerDao.findReaderById(rdId).getRdBorrowQty()!=0){

rdStatus=readerDao.findReaderById(rdId).getRdStatus();

}else{

rdStatus=request.getParameter("rdStatus");//证件状态

}

int rdAdminRoles=readerDao.findReaderById(rdId).getRdAdminRoles();

String getRdPhoto=request.getParameter("rdPhoto");//证件照片

String rdPhoto=rdId+".jpg";

String myRdPhoto= getServletContext().getRealPath("/")+"/libraryImage/reader/"+rdId+".jpg";

copyImge(getRdPhoto, myRdPhoto);//图书封面照片另存

int rdBorrowQty=readerDao.findReaderById(rdId).getRdBorrowQty();//已借书数量

String rdPwd=readerDao.findReaderById(rdId).getRdPwd();//读者密码

TB\_Reader reader=new TB\_Reader(rdId, rdName, rdSex, rdType, rdDept, rdPhone, rdEmail, rdDateReg, rdPhoto, rdStatus, rdBorrowQty, rdPwd, rdAdminRoles);

readerDao.updateReader(reader);

response.sendRedirect("findReader.jsp");

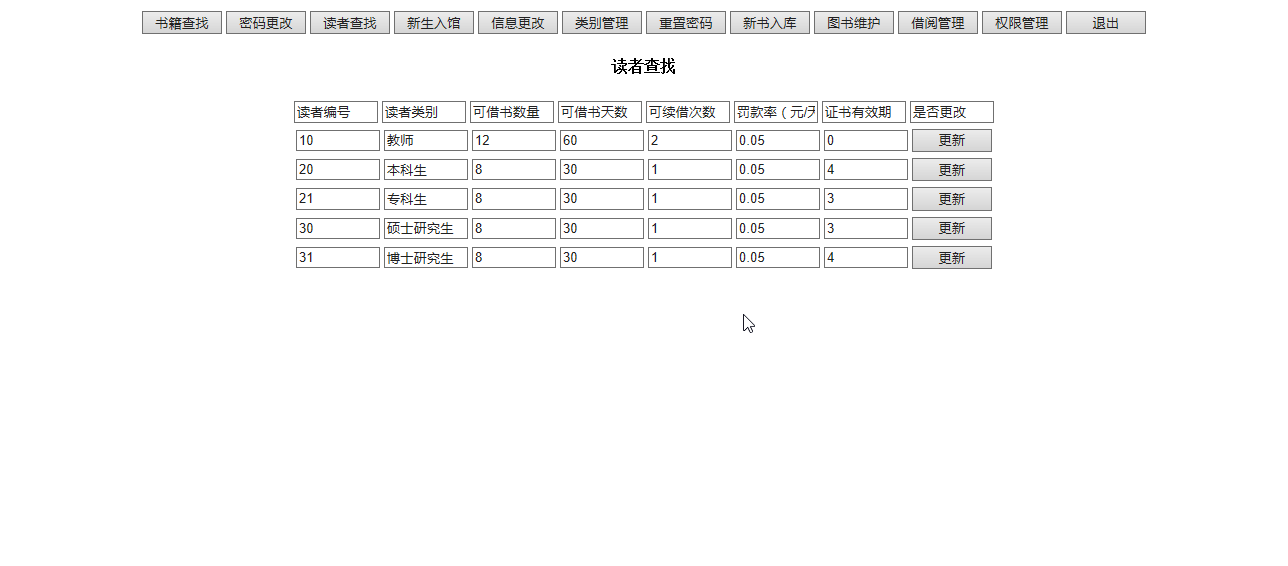
} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

#### 类别管理



输出所有读者类别，并可更该读者类别对应的信息，点击更新，更改的信息将被写入数据库保存

页面实现：

<div class="main" align="center">

<table width="300px">

<tr>

<td><input value="读者编号" readonly="readonly"/></td>

<td><input value="读者类别" readonly="readonly"/></td>

<td><input value="可借书数量" readonly="readonly"/></td>

<td><input value="可借书天数" readonly="readonly"/></td>

<td><input value="可续借次数" readonly="readonly"/></td>

<td><input value="罚款率（元/天）" readonly="readonly"/></td>

<td><input value="证书有效期" readonly="readonly"/></td>

<td><input value="是否更改" readonly="readonly"/></td>

</tr>

</table>

<c:forEach items="${typeList}" var="items">

<form action="UpdateReaderTypeServlet" method="post">

<table>

<tr>

<td><input type="text" name="rdType" value="${items.getRdType() }" readonly="readonly" /></td>

<td ><input type="text" name="typeListName${items.getRdType() }" value="${items.getRdTypeName() }" readonly="readonly"/></td>

<td><input type="text"name="typeListQty${items.getRdType() }" value="${items.getCanLendQty() }"/></td>

<td><input type="text"name="typeListDay${items.getRdType() }" value="${items.getCanLendDay()}"/></td>

<td><input type="text" name="typeListTimes${items.getRdType() }" value="${items.getCanContinueTimes() }"/></td>

<td><input type="text" name="typeListRate${items.getRdType() }" value="${items.getPunishRate() }"/></td>

<td><input type="text" name="typeListDate${items.getRdType() }" value="${items.getDateValid() }"/></td>

<td><input type="submit" value="更新"/></td>

</tr>

</table>

</form>

</c:forEach>

<c:if test="${updateType!=null }">

<c:out value="${updateType }"></c:out>

</c:if>

</div>

在输出所有读者类别与更新之前需要先查找数据类别，查找功能实现代码如下：

try {

HttpSession session=request.getSession();

TB\_ReaderTypeDao typeDao=new TB\_ReaderTypeDao();

List<TB\_ReaderType> typeList=typeDao.findAllType();

session.setAttribute("typeList", typeList);

response.sendRedirect("findReaderType.jsp");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

当需要更新读者类别信息时，更新功能的后台实现如下：

try {

request.setCharacterEncoding("UTF-8");

String myRdType=request.getParameter("rdType");

int rdType=Integer.parseInt(myRdType);

String myTypeListName="typeListName"+myRdType;

String myTypeListQty="typeListQty"+rdType;

String myTypeListDay="typeListDay"+rdType;

String myTypeListTimes="typeListTimes"+rdType;

String myTypeListRate="typeListRate"+rdType;

String myTypeListDate="typeListDate"+rdType;

String rdTypeName=request.getParameter(myTypeListName);//读者名称

String myCanLendQty=request.getParameter(myTypeListQty);

int canLendQty=Integer.parseInt(myCanLendQty);//可借书数量

String myCanLendDay=request.getParameter(myTypeListDay);

int canLendDay=Integer.parseInt(myCanLendDay);//可借书天数

String myCanContinueTimes=request.getParameter(myTypeListTimes);

int canContinueTimes=Integer.parseInt(myCanContinueTimes);//可续借次数

String myPunishRate=request.getParameter(myTypeListRate);

float punishRate=Float.parseFloat(myPunishRate);//罚款率

String myDateValid=request.getParameter(myTypeListDate);

int dateValid=Integer.parseInt(myDateValid);//证书有效期

TB\_ReaderType type=new TB\_ReaderType(rdType, rdTypeName, canLendQty, canLendDay, canContinueTimes, punishRate, dateValid);

TB\_ReaderTypeDao typeDao=new TB\_ReaderTypeDao();

int i=typeDao.updateType(type);

if(i!=0){

request.getSession().setAttribute("updateType","更新成功");

response.sendRedirect("FindReaderTypeServlet");

}else{

System.out.println("更改失败");

}

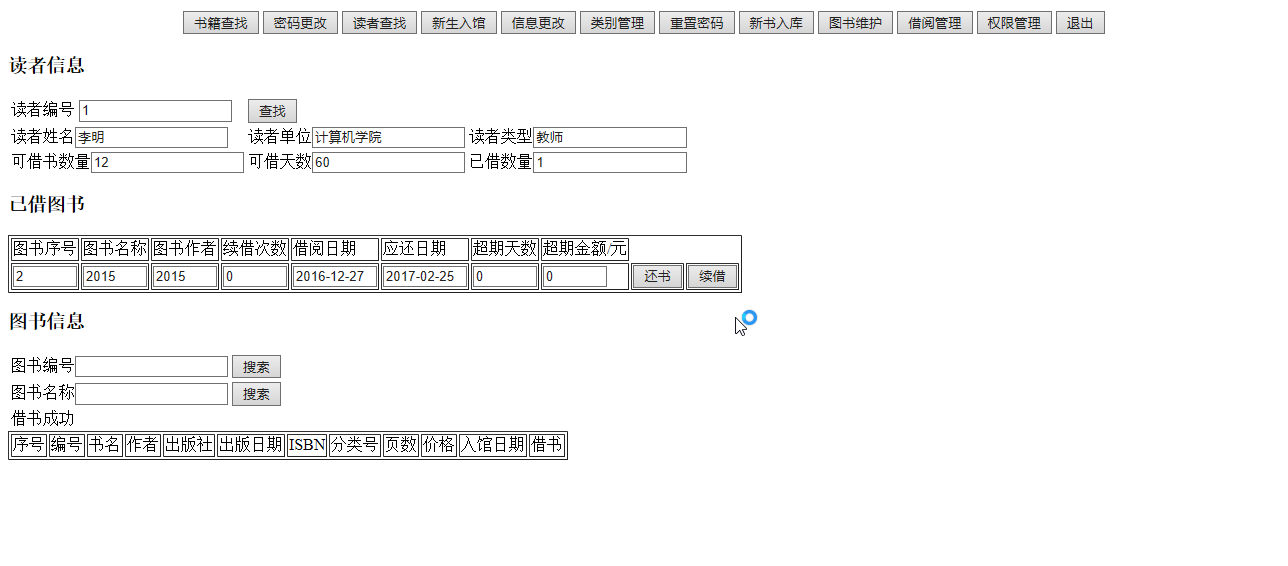
} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

#### 借阅管理



此界面只有在系统管理员和借阅管理员登录时显示。此界面查找了当前读者的所有已借未还书籍，以及超期时间和应罚款金额，并可续借以及借书。页面分为两块，图书信息之上为一块，主要实现续借与还书功能，显示了读者的已借未还书籍和罚款信息，第二块为搜索图书并实现借书功能。

第一块为输出读者的信息以及借书未还数量并提供续借和还书的功能，此块界面实现为：

<form action="BorrowFindReaderServlet" method="post">

<h3>读者信息</h3>

<table>

<tr>

<td>读者编号

<input type="text" name="rdId" value="${reader.getRdId()}"/>

</td>

<td><input type="submit" value="查找"/></td>

</tr>

<tr>

<td>读者姓名<input type="text" readonly="readonly" value="${reader.getRdName() }"/></td>

<td>读者单位<input type="text" readonly="readonly" value="${reader.getRdDept() }"/></td>

<td>读者类型<input type="text" readonly="readonly" value="${type.getRdTypeName()}"/></td>

</tr>

<tr>

<td>可借书数量<input type="text" readonly="readonly" value="${type.getCanLendQty()}"/></td>

<td>可借天数<input type="text" readonly="readonly" value="${type.getCanLendDay()}"/></td>

<td>已借数量<input type="text" readonly="readonly" value="${borrowList.size()}"/></td>

</tr>

</table>

<c:if test="${nextBorrowError!=null }">

<c:out value="${nextBorrowError }"></c:out>

<c:remove var="nextBorrowError" scope="session"/>

</c:if>

</form>

<h3>已借图书</h3>

<table border="1">

<tr>

<td>图书序号</td>

<td>图书名称</td>

<td>图书作者</td>

<td>续借次数</td>

<td>借阅日期</td>

<td>应还日期</td>

<td>超期天数</td>

<td>超期金额/元</td>

</tr>

<c:forEach items="${borrowList}" var="items" varStatus="loop">

<tr>

<td><input type="text" style="width:60px;"value="${items.getBkID() }"/></td>

<td><input type="text" style="width:60px;"value="${ReturnBookList[loop.count-1].getBkName()}"/></td>

<td><input type="text" style="width:60px;"value="${ReturnBookList[loop.count-1].getBkAuthor()}"/></td>

<td><input type="text" style="width:60px;"value="${items.getIdContinueTimes() }"/></td>

<td><input type="text"style="width:80px;" value="${items.getIdDateOut()}"/></td>

<td><input type="text" style="width:80px;"value="${items.getIdDateRetPlan() }"/></td>

<td><input type="text" style="width:60px;"value="${items.getIdOverDay() }"/></td>

<td><input type="text" style="width:60px;"value="${items.getIdOverMoney() }"/></td>

<td><a href="ReturnBookServlet?bkId=${items.getBkID()}"><input type="button" value="还书"/></a></td>

<td><a href="NextBorrowBookServlet?bkId=${items.getBkID()}"><input type="button" value="续借"/></a></td>

</tr>

</c:forEach>

</table>

</div>

<div>

第二块为查找书籍并提供借书功能，界面实现为：

<form action="BorrowFindBookServlet" method="post">

<h3>图书信息</h3>

<table>

<tr>

<td>图书编号<input type="text" name="bkId" value="${BorrowBkId}"/></td>

<td><input type="submit" value="搜索"/></td>

</tr>

<tr>

<td>图书名称<input type="text" name="bkName" value="${BorrowBkName }"/></td>

<td><input type="submit" value="搜索"/></td>

</tr>

<c:if test="${findBorrowError!=null }">

<tr><td><c:out value="${findBorrowError }"></c:out> </td></tr>

<c:remove var="findBorrowError" scope="session"/>

</c:if>

<c:if test="${borrowError!=null }">

<tr><td><c:out value="${borrowError }"></c:out> </td></tr>

<c:remove var="borrowError" scope="session"/>

</c:if>

</table>

</form>

<table border="1" >

<tr>

<td>序号</td>

<td>编号</td>

<td>书名</td>

<td>作者</td>

<td>出版社</td>

<td>出版日期</td>

<td>ISBN</td>

<td>分类号</td>

<td>页数</td>

<td>价格</td>

<td>入馆日期</td>

<td>借书</td>

</tr>

<c:forEach items="${BorrowBookList}" var="items">

<tr>

<td><input type="text" style="width:60px;" value="${items.getBkId()}" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkCode() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkName() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkAuthor() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkPress() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkDatePress() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkISBN() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkCatalog() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkPages() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkPrice() }" readonly="readonly"/></td>

<td><input type="text" style="width:60px;"value="${items.getBkDateIn() }" readonly="readonly"/></td>

<td><a href="BorrowBookServlet?bkId=${items.getBkId()}" ><input type="button" value="借书"/></a></td>

</tr>

</c:forEach>

<c:remove var="BorrowBookList" scope="session"/>

</table>

</div>

第一界面中需要查找读者信息并输出读者已借未还书籍以及惩罚的功能实现为：

HttpSession session=request.getSession();

try {

String myRdId=request.getParameter("rdId");

if(myRdId!=null){

int rdId=Integer.parseInt(myRdId);

session.setAttribute("BorrowRdId", rdId);

}

TB\_ReaderDao readerDao=new TB\_ReaderDao();

TB\_Reader reader= readerDao.findReaderById((int) session.getAttribute("BorrowRdId"));

TB\_ReaderTypeDao typeDao=new TB\_ReaderTypeDao();

TB\_ReaderType type=typeDao.findType(reader.getRdType());

TB\_BorrowDao borrowDao=new TB\_BorrowDao();

List<TB\_Borrow> borrowList=new ArrayList<TB\_Borrow>();

borrowList=borrowDao.findAllBorrow((int) session.getAttribute("BorrowRdId"));

TB\_BookDao bookDao=new TB\_BookDao();

List<TB\_Book> bookList=new ArrayList<TB\_Book>();

Date nowDate=new Date(System.currentTimeMillis());

for(int i=0;i<borrowList.size();i++){

if(!borrowList.get(i).isHasReturn()){

int days=differDays(nowDate,borrowList.get(i).getIdDateRetPlan());

if(days<=0){

borrowList.get(i).setIdOverDay(0);

borrowList.get(i).setIdOverMoney(BigDecimal.ZERO);

}else{

borrowList.get(i).setIdOverDay(days);

borrowList.get(i).setIdOverMoney(BigDecimal.valueOf(borrowList.get(i).getIdOverDay()\*type.getPunishRate()));

}

TB\_Book book=bookDao.findBookById(borrowList.get(i).getBkID());

bookList.add(book);

}

}

session.setAttribute("reader", reader);

session.setAttribute("type", type);

session.setAttribute("borrowList", borrowList);

session.setAttribute("ReturnBookList", bookList);

response.sendRedirect("borrowBook.jsp");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

续借功能的实现为：

HttpSession session=request.getSession();

try {

String myBkId=request.getParameter("bkId");

int bkId=Integer.parseInt(myBkId);

//

TB\_BorrowDao borrowDao=new TB\_BorrowDao();

TB\_Borrow borrow=borrowDao.findBookBorrow(bkId);

TB\_ReaderDao readerDao=new TB\_ReaderDao();

TB\_Reader reader=readerDao.findReaderById((int)session.getAttribute("BorrowRdId"));

if(reader.getRdStatus().equals("有效")){

TB\_ReaderTypeDao typeDao=new TB\_ReaderTypeDao();

TB\_ReaderType type= typeDao.findType(reader.getRdType());

Date idDateRetPlan=plusDate(borrow.getIdDateRetPlan(), type.getCanLendDay());

borrow.setIdDateRetPlan(idDateRetPlan);

String operatorLend=(String)session.getAttribute("admin");

borrow.setOperatorLend(operatorLend);

if(type.getCanContinueTimes()>borrow.getIdContinueTimes()){

int i=borrowDao.nextBorrow(borrow);

if(i!=0){

session.setAttribute("nextBorrowError", "续借成功");

}else{

session.setAttribute("nextBorrowError", "续借失败");

}

}else{

session.setAttribute("nextBorrowError", "续借失败");

}

}else{

session.setAttribute("nextBorrowError", "借书证处于无法使用状态");

}

response.sendRedirect("BorrowFindReaderServlet");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

还书功能实现为：

HttpSession session=request.getSession();

try {

String myBkId=request.getParameter("bkId");

int bkId=Integer.parseInt(myBkId);

TB\_BookDao bookDao=new TB\_BookDao();

TB\_Book book=bookDao.findBookById(bkId);

book.setBkStatus("在馆");

bookDao.changBook(book);

//

TB\_BorrowDao borrowDao=new TB\_BorrowDao();

TB\_Borrow borrow=borrowDao.findBookBorrow(bkId);

borrow.setHasReturn(true);

borrow.setIdDateRetAct(new Date(System.currentTimeMillis()));

TB\_ReaderDao readerDao=new TB\_ReaderDao();

TB\_Reader reader=readerDao.findReaderById((int)session.getAttribute("BorrowRdId"));

int days=differDays(borrow.getIdDateRetAct(),borrow.getIdDateRetPlan());

TB\_ReaderTypeDao typeDao=new TB\_ReaderTypeDao();

TB\_ReaderType type= typeDao.findType(reader.getRdType());

if(days<=0){

borrow.setIdOverDay(0);

borrow.setIdOverMoney(BigDecimal.ZERO);

}else{

borrow.setIdOverDay(days);

borrow.setIdOverMoney(BigDecimal.valueOf(borrow.getIdOverDay()\*type.getPunishRate()));

}

String operatorLend=String.*valueOf*((**int**)session.getAttribute("admin"));

borrow.setOperatorLend(operatorLend);//还书管理员

borrowDao.changBorrow(borrow);

int rdBorrowQty=reader.getRdBorrowQty()-1;

reader.setRdBorrowQty(rdBorrowQty);

response.sendRedirect("BorrowFindReaderServlet");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

第二块界面中查找书籍的功能有两种模式，一种根据图书编号，第二种是根据图书名称，但可以用一个后台实现，代码实现为：

HttpSession session=request.getSession();

try {

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

String myBkId=request.getParameter("bkId");

session.setAttribute("BorrowBkId", myBkId);

List<TB\_Book> BorrowBookList=new ArrayList<TB\_Book>();

if(myBkId==null||myBkId.equals("")){

List<String> column=new ArrayList<String>();

column.add("bkName");

List<String> info=new ArrayList<String>();

info.add(bkName);

TB\_BookDao bookDao=new TB\_BookDao();

BorrowBookList=bookDao.findBook(column, info);

}else{

int bkId=Integer.parseInt((String)session.getAttribute("BorrowBkId"));

TB\_BookDao bookDao=new TB\_BookDao();

TB\_Book book=bookDao.findBookById(bkId);

BorrowBookList.add(book);

}

//判断书是否在馆

for(int i=0;i<BorrowBookList.size();i++){

if(BorrowBookList.get(i)!=null){

Boolean bool=BorrowBookList.get(i).getBkStatus().equals("在馆");

if(!bool){

BorrowBookList.remove(i);

i--;

}

session.setAttribute("BorrowBookList", BorrowBookList);

}else{

session.setAttribute("findBorrowError", "此书不在馆");

}

}

session.setAttribute("BorrowBkName", bkName);

response.sendRedirect("BorrowFindReaderServlet");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

借书功能实现为：

HttpSession session=request.getSession();

try {

String myBkId=request.getParameter("bkId");

int rdId=(Integer) session.getAttribute("BorrowRdId");

int bkId=Integer.parseInt(myBkId);

// int rdId=Integer.parseInt(myRdId);

TB\_BookDao bookDao=new TB\_BookDao();

TB\_Book book=bookDao.findBookById(bkId);

book.setBkStatus("借出");

//

TB\_ReaderDao readerDao=new TB\_ReaderDao();

TB\_Reader reader=readerDao.findReaderById(rdId);

if(reader.getRdStatus().equals("有效")){

int rdBorrowQty=reader.getRdBorrowQty()+1;

reader.setRdBorrowQty(rdBorrowQty);

//获取应还书日期

TB\_ReaderTypeDao typeDao=new TB\_ReaderTypeDao();

TB\_ReaderType type=typeDao.findType(reader.getRdType());

TB\_BorrowDao borrowDao=new TB\_BorrowDao();

Date idDateOut=new Date(System.currentTimeMillis());

Date idDateRetPlan=plusDate(idDateOut, type.getCanLendDay());

int i=(int)session.getAttribute("admin");

String operatorLend=String.valueOf(i);

TB\_Borrow borrow=new TB\_Borrow(rdId, bkId, 0, idDateOut, idDateRetPlan, false,operatorLend);

borrowDao.insertBorrow(borrow);//插入借书数据

readerDao.updateReader(reader);//读者借书量加一

bookDao.changBook(book);//将书籍状态设置为借出

session.setAttribute("borrowError", "借书成功");

}else{

session.setAttribute("borrowError", "借书失败");

}

response.sendRedirect("BorrowFindBookServlet");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

#### 权限管理



此界面显示于系统管理员登录时，此界面功能在于更改读者权限，此界面搜索读者编号后显示其对应管理员类型，然后选择管理员类型后点击更新，即可更新其管理员类型

页面实现：

<form action="FindAdminRolesServlet" method="post">

<table>

<tr>

<td>编号</td>

<td><input type="text" name="rdId" value="${adminRdId }"/></td>

<td><input type="submit" value="搜索" /></td>

</tr>

</table>

</form>

<form action="UpdateAdminRolesServlet" method="post">

<table>

<tr>

<td>管理员类型</td>

<td><select name="rdAdminRoles" id="rdAdminRoles">

<option value="0">系统管理</option>

<option value="1">读者管理</option>

<option value="2">图书管理</option>

<option value="3">借书管理</option>

<option value="4">读者</option>

</select>

</td>

<c:if test="${AdminRoles!=null }">

<td><input type="submit" value="更改管理员类型"/></td>

</c:if>

</tr>

</table>

</form>

<script type="text/javascript">

for(var i=0;i<document.getElementById('rdAdminRoles').options.length;i++){

if(document.getElementById('rdAdminRoles').options[i].value=='${AdminRoles}'){

document.getElementById('rdAdminRoles').options[i].selected=true;

}

}

</script>

页面有搜索和更新两个功能，搜索功能实现为：

try {

HttpSession session=request.getSession();

String myRdId=request.getParameter("rdId");

int rdId=Integer.parseInt(myRdId);

TB\_ReaderDao readerDao=new TB\_ReaderDao();

TB\_Reader reader=readerDao.findReaderById(rdId);

session.setAttribute("adminRdId", rdId);

session.setAttribute("AdminRoles", reader.getRdAdminRoles());

response.sendRedirect("adminRoles.jsp");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

更改功能实现为：

try {

HttpSession session=request.getSession();

String myRdAdminRoles=request.getParameter("rdAdminRoles");

int rdAdminRoles=Integer.parseInt(myRdAdminRoles);

int rdId=(int) session.getAttribute("adminRdId");

TB\_ReaderDao readerDao=new TB\_ReaderDao();

TB\_Reader reader=readerDao.findReaderById(rdId);

reader.setRdAdminRoles(rdAdminRoles);

int i=readerDao.updateReader(reader);

if(i!=0){

session.setAttribute("adminRolesBoolean","更改成功!");

//管理部分页面设置

session.setAttribute("AdminRoles", reader.getRdAdminRoles());

response.sendRedirect("adminRoles.jsp");

}else{

session.setAttribute("adminRolesBoolean","更改失败!");

}

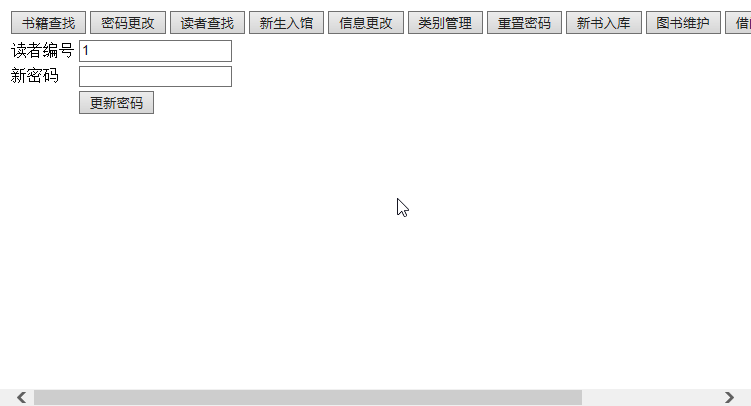
} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

#### 密码更改



密码更改此功能适用于所有用户，此块功能为更新本人密码，只需输入新密码即可更改密码

页面实现：

<form action="UpdatePasswordServlet" method="post">

<table>

<tr>

<td>读者编号</td>

<td><input type="text" name="rdId" value="${admin }" readonly="readonly"/></td>

</tr>

<tr>

<td>新密码</td>

<td><input type="text" name="rdPwd"/></td>

</tr>

<tr>

<td><input type="hidden" name="passType" value="my"/></td>

<td><input type="submit" value="更新密码"/></td>

</tr>

<tr>

<c:if test="${myPassword!=null }">

<c:out value="${myPassword }"></c:out>

<c:remove var="myPassword" scope="session" />

</c:if>

</tr>

</table>

</form>

密码更新功能只需将读者对应数据库密码更改即可，代码实现为:

try {

HttpSession session=request.getSession();

String myRdId=request.getParameter("rdId");

int rdId=Integer.parseInt(myRdId);

int passType=(int) session.getAttribute("admin");

TB\_ReaderDao readerDao=new TB\_ReaderDao();

TB\_Reader reader=readerDao.findReaderById(rdId);

if(rdId==passType){

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

reader.setRdPwd(rdPwd);

}else{

String rdPwd="123";

reader.setRdPwd(rdPwd);

}

readerDao.updateReader(reader);

if(rdId==passType){

session.setAttribute("myPassword", "成功更改");

response.sendRedirect("myPassword.jsp");

}else{

session.setAttribute("Password", "重置成功");

response.sendRedirect("password.jsp");

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

#### 重置密码



此界面显示于系统管理员与读者管理员登录时，在于重置忘记密码的读者密码，重置密码为123

页面实现：

<form action="UpdatePasswordServlet" method="post">

<table>

<tr>

<td>编号</td>

<td><input name="rdId" type="text"/></td>

</tr>

<tr>

<td><input name="passType" value="other" type="hidden"/></td>

<td><input type="submit" value="重置密码"/></td>

</tr>

<tr>

<c:if test="${Password!=null }">

<c:out value="${Password }"></c:out>

<c:remove var="Password" scope="session" />

</c:if>

</tr>

</table>

</form>

重置密码的功能实现代码与密码更改公用的一个servlet，在后台会识别是哪个页面导入的数据，然后实现不同的功能

## 6. 课程设计小结

### 个人小结

对于本次课设总体功能的实现对于我而言并不困难，借阅、图书维护、新书入库、新生入馆、信息更新、查找等功能都有很好的实现，然而对于前端的编写对于我而言是一个很大的缺点，需要好好的补一下，还有后端，总结后发现自己有很多不必要重复的代码，如果使用接口来实现会使代码更加精简，使代码的不必要重复减少许多。

### 指导老师意见

成绩: 教师签名：

年 月 日