**内容仅供参考，禁止一切商业行为！**



**数据库原理及应用课程设计报告**

**院 （系）：**  计算机科学学院

**专业班级：**

**姓 名：**

**学 号：**

**指导教师：**

**设计时间：** 2020.11.26 – 2020.12.10

**设计地点：**

目 录

[一、设计目的 1](#_Toc58971636)

[二、设计任务 1](#_Toc58971637)

[三、系统需求 1](#_Toc58971638)

[系统边界与约定 1](#_Toc58971639)

[需求概要 1](#_Toc58971640)

[系统分析目标 2](#_Toc58971641)

[3.1 读者管理 2](#_Toc58971642)

[3.2 图书管理 4](#_Toc58971643)

[3.3 借阅管理 5](#_Toc58971644)

[3.4 用户登录与用户管理 7](#_Toc58971645)

[3.5 读者查询与预约借书\* 8](#_Toc58971646)

[3.6 统计与报表\* 8](#_Toc58971647)

[3.7 数据备份\* 9](#_Toc58971648)

[3.8 系统功能与用户角色 9](#_Toc58971649)

[3.9 领域模型 9](#_Toc58971650)

[四、系统设计 10](#_Toc58971651)

[4.1 数据库设计 10](#_Toc58971652)

[4.2 三层架构简介 12](#_Toc58971653)

[4.4 类总体设计 13](#_Toc58971654)

[4.5 Model层实体类设计 14](#_Toc58971655)

[4.6 DAL层类设计 14](#_Toc58971656)

[4.7 BLL层类设计 14](#_Toc58971657)

[4.8 UI层窗体设计 14](#_Toc58971658)

[五、系统实现 21](#_Toc58971659)

[5.1 数据库设计 21](#_Toc58971660)

[5.3 搭建三层架构解决方案 26](#_Toc58971661)

[5.4 Model层 29](#_Toc58971662)

[4.6 DAL层类 34](#_Toc58971663)

[4.7 BLL层类设计 59](#_Toc58971664)

[4.8 UI层类设计 67](#_Toc58971665)

[六、系统演示 122](#_Toc58971666)

[6.1 登录系统 122](#_Toc58971667)

[6.2 添加35本《c程序设计》 123](#_Toc58971668)

[6.3 维护52号图书 125](#_Toc58971669)

[6.4 借阅图书 127](#_Toc58971670)

[6.5 续借图书 129](#_Toc58971671)

[6.6 归还图书 130](#_Toc58971672)

[6.7 权限管理 131](#_Toc58971673)

[6.8 密码修改 132](#_Toc58971674)

[七、课程设计小结 134](#_Toc58971675)

## 一、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。

## 二、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：

（1）在MySQL 8.0（或更高版本）环境下建立图书管理系统所使用的数据库，利用SSMS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；

（2）掌握ADO.NET编程技术，对MySQL数据库进行连接和操纵；

（3）掌握使用C#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。

（4）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。

## 三、系统需求

### 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；其它功能需求包括：读者查询与预约借书\*、统计与报表\*、数据备份\*、书架管理\*、期刊管理\*、期刊借阅\*、论文管理\*、论文借阅\*等。（\* 表示留待以后的迭代周期完成，下同）

（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。

### 系统分析目标

找出系统用例，书写用例文档，建立领域模型（概念模型）。

[用例] use case，或译为使用案例、用况。是指为了完成一个领域目标或任务，提供一个或多个场景来说明其在系统内部与外部（人员或外部系统等）之间的交互过程。如办理借书证、借书、还书、用户登录等均为用例；而借阅管理不是用例，因为它包括多个任务；读者可否借书也不是用例，因为它不是一个领域任务或业务目标。参见“参考资料01 用例.doc”。

用例是一个事件流，一个事件对应一个系统操作；如借书用例中的“根据借书证号查询读者信息”、“根据借书证号查询超期未归还图书”、“判断读者可否借书”、“判断图书是否在馆”、“确认借书”等均为事件或系统操作。

找出系统用例后，对每个用例的交互操作过程进行描述（即书写用例），通过用例发现领域概念及其属性，并建立领域模型（即概念模型）。用例与领域模型的建立过程一般是交互进行的，并相互参考、印证和补充完善。

通过用例发现事件或系统操作，可进行下一步的设计和实现工作，是系统设计主要内容。

在面向对象方法的开发过程中，分析与设计没有明显的界限，只是侧重面不同而已。

### 3.1 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、批量办理新生借书证\*等。（\*表示留待迭代2及以后完成，下同。完成此类用例文档并实现其功能者，教师可酌情加分）

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。 |

**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（略，要求自己完成）。

**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。系统修改读者状态为挂失。 |
| 后置条件：系统记录读者信息。 |
| 业务规则：借书证挂失后禁止读者借书。 |

分析：修改读者概念，增加属性“借书证状态”；并修改办理借书证用例。

**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。 |
| 5. 确认解除挂失。（前提：读者状态为挂失） |
| 6. 系统修改读者状态为有效。 |
| 后置条件：系统记录读者信息。 |

**5. 补办借书证\***

补办借书证过程：创建新借书证（复制原读者信息，但借书证需取新号，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。

**6. 注销借书证\***

相关规则：有未归还图书者不能办理注销。

**7. 批量办理新生借书证\***

从教务管理系统导入新生信息，系统生成借书证信息，管理员核实后办理借书证。

**8. 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。

### 3.2 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

图书分类概念暂不考虑,可留待以后开发周期处理。

**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. 输入或系统生成起始书号(BkNO)。 |
| 2. 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。 |

馆藏图书目录=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、入库数量、入馆日期

图书=书号、分类号、图书状态

**3. 图书信息维护**

图书信息的查、删、改等信息维护操作。

**4. 在馆图书变卖与销毁处理\***

长期没有借阅记录且失去保存价值的图书可以进行变卖或销毁处理。

### 3.3 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。

（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”。 |
| 2. 系统查询读者信息、未归还图书信息（含超期），并进行显示。 |
| 3. 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。 |
| 4. 管理员输入待借图书的书号，点击“查询图书”。 |
| 5. 系统查询，并显示图书信息。 |
| 6. 系统判断图书是否在馆，若不在馆，则禁止借书。 |
| 7. 管理员点击“借书”。 |
| 8. 系统完成以下操作： |
| （1）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； |
| （2）修改读者对象的已借书数量+1； |
| （3）修改图书状态为借出； |
| 后置条件：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。 |

修改读者信息：增加“已借书数量”；修改用例办理借书证：后置条件中，已借书数量=0。

**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。 |
| 2. 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。 |
| 3. 系统判断可否续借（续借次数<可续借次数, 读者状态为有效）。 |
| 4. 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。

### 3.4 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：可采用4位二进制，借书证管理(0001)2=1、图书管理(0010)2=2、借阅管理(0100)2=4、系统管理(1000)2=8。表示图书管理和借阅管理权限：2+4=6；判断7是否具备图书管理权限：7位与2，即(0111)2位与(0010)2=(0010)2，表示有此权限。

分析：将管理员信息合并到读者中，可简化概念，此时，非管理员的管理角色(0000)2=0。好处：简化用户登录用例；坏处：概念不够清晰，给用户管理带来不便。第1个迭代周期采用简化概念设计。另外也可用4个逻辑数据项分别表示4个管理角色。

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“登录”。 |
| 3. 系统根据借书证号获取读者对象。 |
| 3a. 未获取到读者对象时，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。 |

**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 |

### 3.5 读者查询与预约借书\*

读者（非管理员用户）的功能需求，包括用例：未归还图书查询与续借\*（含超期、即将到期查询操作）、预约借书\*（含在馆图书查询操作）等。

该类读者所使用的功能最好是采用web程序实现，Web程序功能可包括：读者登录、密码修改、未归还图书查询与续借\*、预约借书\*等。

用例名称：未归还图书查询与续借

用例名称：预约借书（可在迭代2完善借书用例）

### 3.6 统计与报表\*

包括：馆藏图书统计、借阅情况统计、借阅时段统计等。

馆藏图书统计可按图书类别、入馆时间统计图书数量、金额等，可得图书分布情况；

借阅情况统计可按单位（含专业、年级）、图书类别、借阅时间（年、学期、月份）等统计借阅次数和天数，统计结果可作为订购新书的参考依据；

借阅时段统计按工作日（星期一、二、……）和时段（8:00-8:30、8:30-9:00、……）统计借还书次数，可得各时段的借阅工作量，并可依此安排借阅管理人员。

注：涉及OLAP（联机分析处理）技术，开发者需理解“多维数据模型”及相关概念。

### 3.7 数据备份\*

用户备份是对MySQL数据库进行备份，由系统管理员操作。

### 3.8 系统功能与用户角色

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 | 读者预约 | √ | √ | √ | √ | √ |
| 8 | 统计与报表 | √ | √ | √ |  | √ |
| 9 | 数据备份 |  |  |  | √ |  |

### 3.9 领域模型

**领域概念小结**

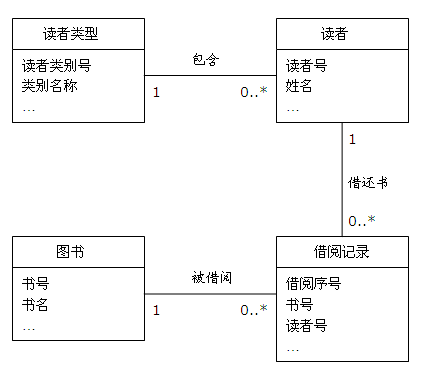
读者=读者号（借书证号）、姓名、性别、所在单位、电话、邮箱地址、办证日期、照片、借书证状态、已借书数量、密码、管理角色。（借书证状态：有效、挂失、注销）

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)**

提示：在领域模型建立后、以及数据库设计完成后，均应检查所有的用例文档（检查重点：事件流、前置条件和后置条件），检查两者的一致性，发现缺漏点及矛盾之处，并进行修正。

## 四、系统设计

### 4.1 数据库设计

#### 1. 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】 |

#### 2. 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(2) | 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 | MediumBlob | 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 |

分析提示：（1）可将管理员另外建表，与读者信息分开单独进行设计。（2）单位rdDept可以采用名称，若采用代码则需另建一张表：单位代码与名称对照表，这样便于按单位统计。（3）管理角色(rdAdminRoles)也可设计为4个bit类型字段。

#### 3. 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 | Decimal(5,2) | 价格 |
| 12 | bkDateIn | DateTime | 入馆日期 |
| 13 | bkBrief | Text | 内容简介 |
| 14 | bkCover | MediumBlob | 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。

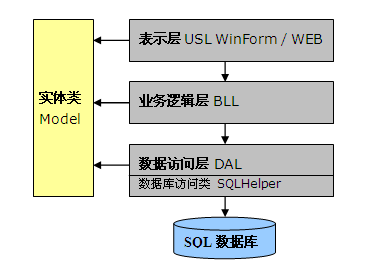
#### 4. 借阅信息表（TB\_Borrow）

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | Decimal(5,2) | 超期金额（应罚款金额） |
| 10 | ldPunishMoney | Decimal(5,2) | 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 |

分析提示：借阅信息即用于图书的借还管理，同时又是统计分析的主要信息来源，故要求保持其历史信息。设计时不能以（读者序号+图书序号）为关键字，否则读者不能第二次借阅同一本书了。

### 4.2 三层架构简介

采用三层体系结构，即表示层、业务逻辑层和数据访问层，如图2所示，图中箭头表示调用和依赖关系。



**图2 三层架构示意图**

**表示层（USL）**：也称**UI**，提供交互式界面，形式：WinForm或HTML Web界面。

**业务逻辑层（BLL）**：实现业务功能，为表示层提供服务，形式：类库。

**数据访问层（DAL）**：实现数据访问功能（如数据库、文件等数据的读取、保存和更新），为业务逻辑层提供服务，形式：类库。微软公布的SQLHelper类提供了对SQL Server数据库的一般访问方法（JAVA有类似的类或组件）。

**实体类（Model）**：描述一个业务实体的类，也即应用系统所涉及的业务对象。对数据库来讲，每个数据表对应于一个实体类，数据表的每个字段对应于类的一个属性。

表示层、业务逻辑层、数据访问层都依赖于业务实体。各层之间数据的传递主要是实体对象，业务信息封装在实体对象中。

### 4.4 类总体设计

实体类、数据访问层类、业务逻辑层类、表示层窗口类的初步设计如表3。

表3 实体类、数据访问层、业务逻辑层的类设计

|  |  |  |
| --- | --- | --- |
| **层** | **类名** | **说明** |
| 实体类模块  （Model） | ReaderType | 读者类型实体类 |
| Reader | 读者实体类 |
| Book | 图书实体类 |
| Borrow | 借阅记录实体类 |
| UserAction | 用户动作实体类 |
| 数据访问层  （DAL） | ReaderTypeDAL | 读者类型数据表访问类（插、删、改、查、存储过程等） |
| ReaderDAL | 读者数据表访问类（插、删、改、查、存储过程等） |
| BookDAL | 图书数据表访问类（插、删、改、查、存储过程等） |
| BorrowDAL | 借阅数据表访问类（插、删、改、查、存储过程等） |
| MySqlHelper | Github开源的对MySQL数据库进行访问的通用类 |
| 业务逻辑层  （BLL） | bookAddControler | 添加图书窗口控制类。 |
| bookDetailControler | 图书详细窗口控制类。 |
| bookSearchControler | 图书搜索窗口控制类。 |
| borrowSearchControler | 借阅图书窗口控制类。 |
| loginControler | 用户登录窗口控制类。 |
| mainControler | 主窗口窗口控制类。 |
| readerSearchControler | 读者搜索窗口控制类。 |
| … | |
| 表示层  （UI） | AboutBox | 关于 窗口类 |
| bookAdd | 添加图书 窗口类 |
| bookDetail | 图书详细/图书修改 窗口类 |
| bookSearch | 图书维护 窗口类 |
| bookSearch\_reader | 借阅图书 窗口类 |
| borrowSearch | 续借/还书 窗口类 |
| changePassword | 修改密码 窗口类 |
| changePermission | 修改权限 窗口类 |
| login | 登录 窗口类 |
| main | 主界面 窗口类 |
| readerSearch | 读者权限管理 窗口类 |
| GlobalObject | 程序数据 共享类 |
| … | |

### 4.5 Model层实体类设计

在keshe.Model项目中添加5个实体类：ReaderType、Reader、Book、Borrow、UserAction，前四个分别对应4个数据库表，而UserAction则是程序中要使用到的，它代表用户的操作。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。

其实实体类就是实现ORM。

ORM Object Relational Mapping 对象关系映射，是为了解决面向对象的类与关系数据库的表之间存在的不匹配的现象，通过使用描述对象和关系之间映射的元数据，在程序中的类对象与关系数据库的表之间建立持久的关系，用于在程序中描述数据库表。本质上就是将数据从一种形式转换到另外一种形式。

ORM是一个广义的概念，适应于关系数据库与应用程序之间的各类数据转换，目前有许多自动转换工具可用，如codesmith 等，也可手工书写实体类代码来实现ORM。

### 4.6 DAL层类设计

除MySqlHelper外，每个数据库表对应一个DAL层类，主要实现该表的插删改查等操作。

### 4.7 BLL层类设计

BLL层类的成员函数设计，可在实现用例的过程中去发现和完成，用例实现主要包括UI层窗口类、BLL层类、DAL层类等方法的设计与实现。

### 4.8 UI层窗体设计

#### 用户登录窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到lable、textbox、button。

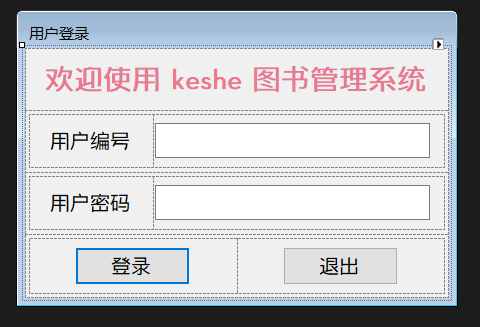


图4.8.1 登录窗口预览

#### 主界面窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到menustrip、datagridview、statusstrip、contextmenustrip。

前四个选项（借书证管理、图书管理、借阅管理、系统管理）视用户身份而显示。

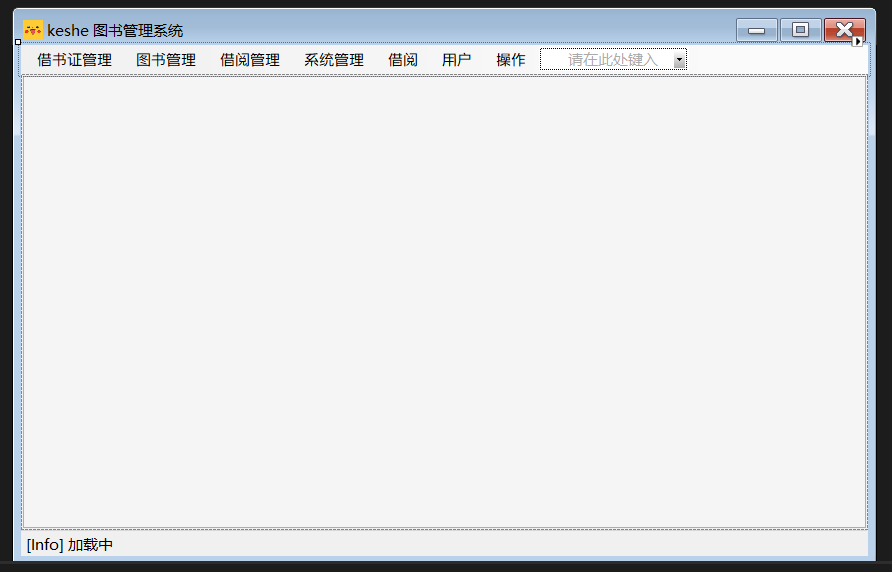


图4.8.2 主窗口预览

#### 添加图书窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到lable、textbox、datetimepicker、combobox、picturebox。

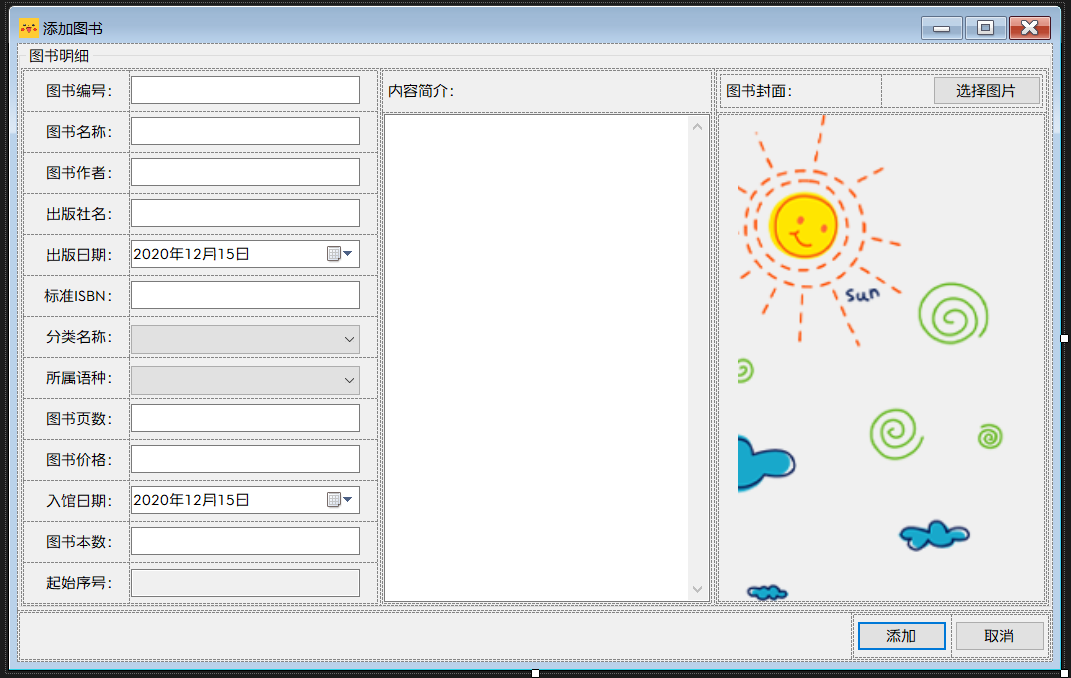


图4.8.3 添加图书窗口预览

#### 图书详细窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到lable、textbox、datetimepicker、combobox、picturebox。

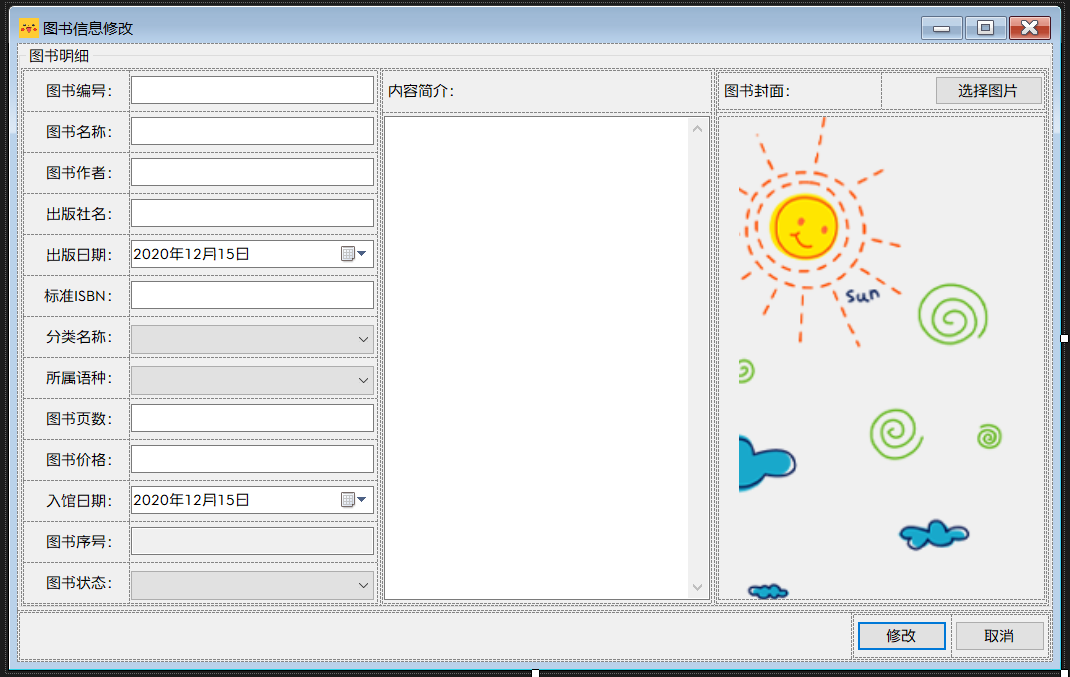


图4.8.4 图书详细窗口预览

#### 图书选择窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到toolstrip、datagridview、statusstrip、contextmenustrip。

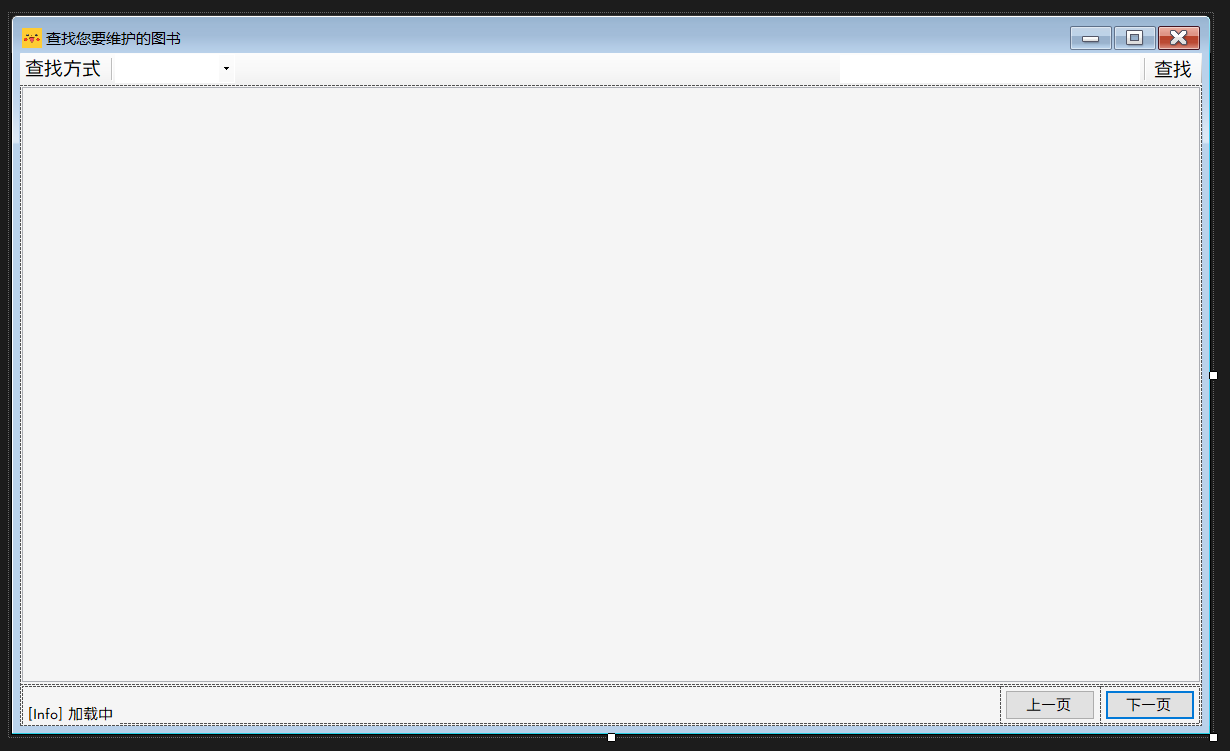


图4.8.5 图书选择窗口预览

#### 借书的图书选择窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到toolstrip、datagridview、statusstrip、contextmenustrip。

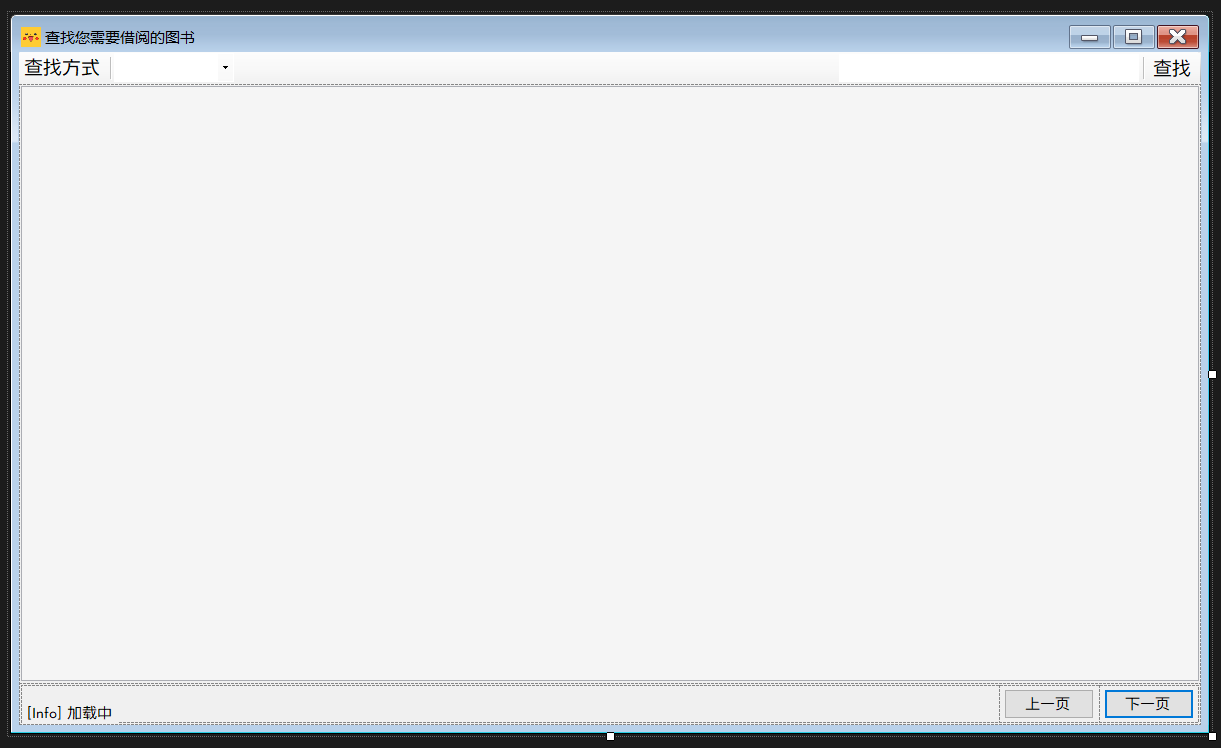


图4.8.6 借书的图书选择窗口预览

#### 续借和还书窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到toolstrip、datagridview、statusstrip、contextmenustrip。

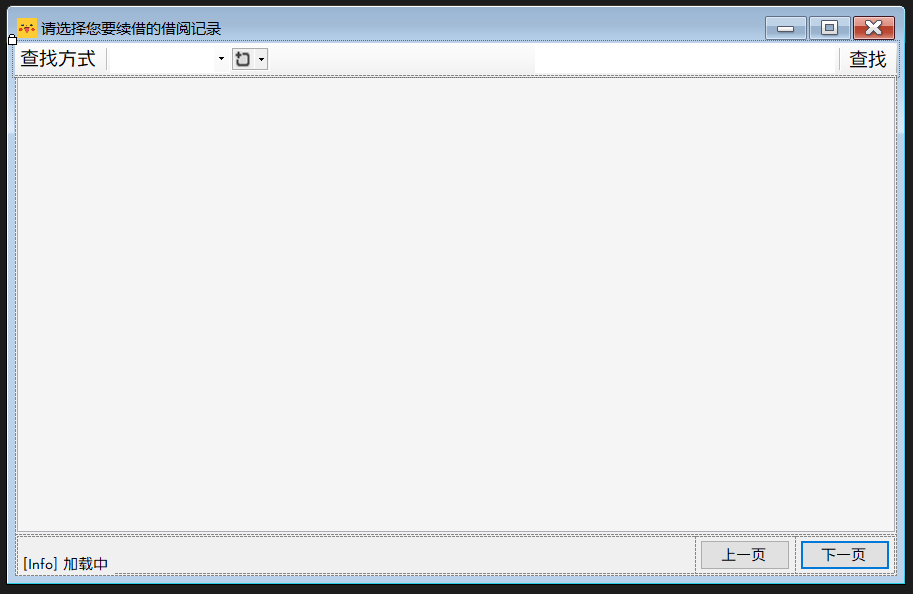


图4.8.7 续借和还书窗口预览

#### 修改密码窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到lable、textbox、button。

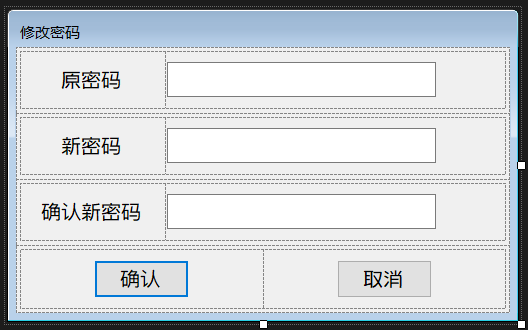


图4.8.8 修改密码窗口预览

#### 修改权限窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到lable、checkbox、button。

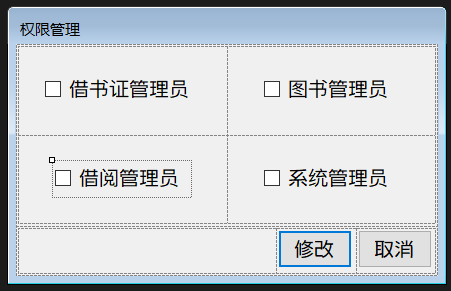


图4.8.9 修改权限窗口预览

#### 管理读者窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到toolstrip、datagridview、statusstrip、contextmenustrip。

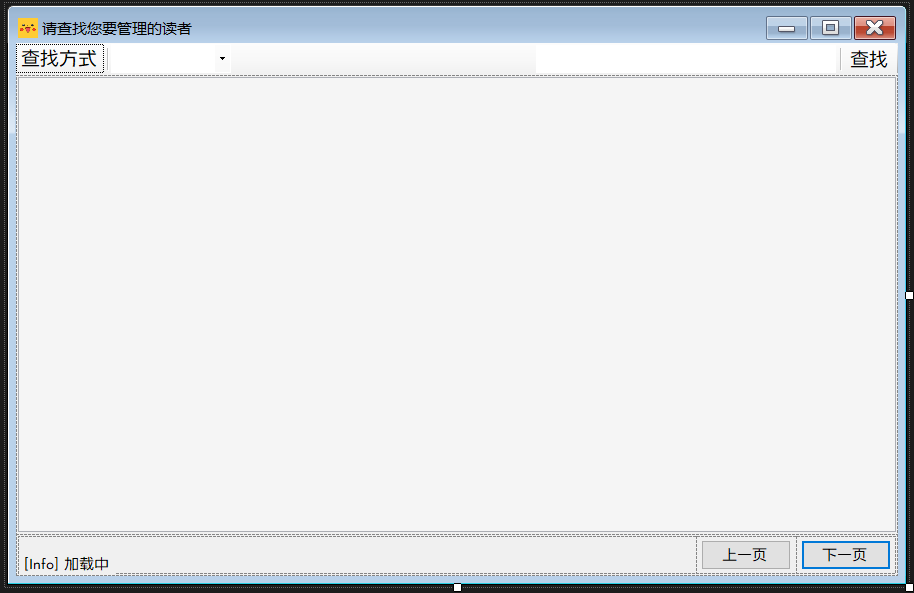


图4.8.10 管理读者窗口预览

#### 关于窗口

使用tableLayoutPanel适应不同大小的窗口，主要用到lable、checkbox、button等。



图4.8.11 关于窗口预览

## 五、系统实现

### 5.1 数据库设计

#### 内容

(1) 创建数据库：keshe；

(2) 使用登录名：root，密码：qweasdwsad；

(3) 创建上述4张数据表及约束；

#### 实现方法

(1) 采用Navicat Premium 15使用SQL代码方式完成创建工作；

(2) 编写SQL脚本文件，并在查询窗口中运行并创建。

#### 1. 读者类别表（TB\_ReaderType）

-- table TB\_ReaderType

create table `TB\_ReaderType` (

`rdType` SmallInt PRIMARY KEY, /\* 读者类别【主键】 \*/

`rdTypeName` Nvarchar(20) unique not null, /\* 读者类别名称【唯一、非空】 \*/

`CanLendQty` int, /\* 可借书数量 \*/

`CanLendDay` int, /\* 可借书天数 \*/

`CanContinueTimes` int, /\* 可续借的次数 \*/

`PunishRate` float, /\* 罚款率（元/天） \*/

`DateValid` SmallInt not null default 0 /\* 证书有效期（年）【非空，0表示永久有效】 \*/

);

#### 2. 读者信息表（TB\_Reader）

-- table TB\_Reader

create table `TB\_Reader` (

`rdID` int PRIMARY KEY, /\* 读者编号/借书证号【主键】 \*/

`rdName` nvarchar(20), /\* 读者姓名 \*/

`rdSex` nchar(2), /\* 性别：男/女 \*/

`rdType` SmallInt not null, /\* 读者类别【外键TB\_ReaderType】 \*/

`rdDept` nvarchar(20), /\* 单位代码/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/办证日期 \*/

`rdPhoto` MediumBlob, /\* 读者照片 \*/

`rdStatus` nchar(2), /\* 证件状态，3个：有效、挂失、注销 \*/

`rdBorrowQty` Int default 0, /\* 已借书数量（缺省值0） \*/

`rdPwd` nchar(32) default '202CB962AC59075B964B07152D234B70', /\* 读者密码（初值123），32bit MD5加密存储 \*/

`rdAdminRoles` SmallInt, /\* 管理角色：(第一位)借书证管理、(第二位)图书管理、(第三位)借阅管理、(第四位)系统管理，可组合 \*/

FOREIGN KEY(`rdType`) REFERENCES `TB\_ReaderType`(`rdType`) /\* 表级约束 \*/

);

#### 3. 图书信息表（TB\_Book）

-- table TB\_Book

CREATE TABLE `TB\_Book` (

`bkID` int PRIMARY KEY, /\* 图书序号【标识列，主键】 \*/

`bkCode` Nvarchar(20), /\* 图书编号或条码号（前文中的书号） \*/

`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（如：TP316-21/123） \*/

`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` DECIMAL(5,2), /\* 价格 \*/

`bkDateIn` DateTime, /\* 入馆日期 \*/

`bkBrief` Text, /\* 内容简介 \*/

`bkCover` MediumBlob, /\* 图书封面照片 \*/

`bkStatus` NChar(2) /\* 图书状态：在馆、借出、遗失、变卖、销毁 \*/

);

#### 4. 借阅信息表（TB\_Borrow）

-- table TB\_Borrow

CREATE TABLE `TB\_Borrow` (

`BorrowID` bigint not null auto\_increment PRIMARY KEY, /\* 借书顺序号【主键】 \*/

`rdID` int, /\* 读者序号【外键TB\_Reader】 \*/

`bkID` int, /\* 图书序号【外键TB\_Book】 \*/

`ldContinueTimes` int default 0, /\* 续借次数（第一次借时，记为0） \*/

`ldDateOut` DateTime, /\* 借书日期 \*/

`ldDateRetPlan` DateTime, /\* 应还日期 \*/

`ldDateRetAct` DateTime, /\* 实际还书日期 \*/

`ldOverDay` Int, /\* 超期天数 \*/

`ldOverMoney` DECIMAL(5,2), /\* 超期金额（应罚款金额） \*/

`ldPunishMoney` DECIMAL(5,2), /\* 罚款金额 \*/

`lsHasReturn` Bit default 0, /\* 是否已经还书，缺省为0-未还 \*/

`OperatorLend` Nvarchar(20), /\* 借书操作员 \*/

`OperatorRet` Nvarchar(20), /\* 还书操作员 \*/

FOREIGN KEY(`rdID`) REFERENCES `TB\_Reader`(`rdID`), /\* 表级约束 \*/

FOREIGN KEY(`bkID`) REFERENCES `TB\_Book`(`bkID`) /\* 表级约束 \*/

)auto\_increment=1;

#### 5. 默认数据

-- default table TB\_Reader

insert into `TB\_Reader` values

(0, '用户已被移除', '男', 0, '', '', '', '2000-01-01', null, '注销', 0, '202CB962AC59075B964B07152D234B70', 0),

(2020001, 'whalechoi', '男', 2, 'Yangtze University', '12345678901', 'cj1369636717@gmail.com', '2018-09-01', null, '有效', 0, '202CB962AC59075B964B07152D234B70', 5),

(2020002, 'tester1', '男', 2, 'Yangtze University', '12345678901', 'tester@gmail.com', '2018-09-01', null, '有效', 0, '202CB962AC59075B964B07152D234B70', 0),

(2020003, 'tester2', '男', 2, 'Yangtze University', '12345678901', 'tester@gmail.com', '2018-09-01', null, '有效', 0, '202CB962AC59075B964B07152D234B70', 0),

(2020004, 'tester3', '男', 2, 'Yangtze University', '12345678901', 'tester@gmail.com', '2018-09-01', null, '有效', 0, '202CB962AC59075B964B07152D234B70', 0);

-- default table TB\_Book

insert into `TB\_Book` values

(0, '', '图书已被移除', '', '', '2000-01-01', '', '', 0, 0, 0.00, '2000-01-01', '', null, '销毁');

#### 6. 触发器

##### TB\_ReaderType

delimiter //

create trigger `TB\_ReaderType` before delete on `TB\_ReaderType`

for each row

begin

update `TB\_Reader` set `rdType` = 0 where `rdType` = old.rdType;

end //

delimiter ;

##### TB\_Reader

delimiter //

create trigger `TB\_Reader` before delete on `TB\_Reader`

for each row

begin

update `TB\_Borrow` set `rdID` = 0 where rdID = old.rdID;

end //

delimiter ;

##### TB\_Book

delimiter //

create trigger `TB\_Book` before delete on `TB\_Book`

for each row

begin

update `TB\_Borrow` set `bkID` = 0, `lsHasReturn` = 1 where bkID = old.bkID;

end //

delimiter ;

#### 6. 存储过程

##### usp\_borrow\_book

delimiter //

create procedure `usp\_borrow\_book` (in inrdID int,in inbkID int,in inOperatorLend varchar(20))

begin

set @CanLendDay=(

select CanLendDay from TB\_ReaderType

where rdType=(

select TB\_Reader.rdType from TB\_Reader where TB\_Reader.rdID=inrdID

)

);

insert into TB\_Borrow values

(NULL, inrdID, inbkID, 0, CURRENT\_DATE, DATE\_ADD(CURRENT\_DATE, interval @CanLendDay day), NULL, NULL, NULL, NULL, 0, inOperatorLend, NULL);

update TB\_Book set bkStatus = '借出' where bkID = inbkID;

end //

delimiter ;

##### usp\_return\_book

delimiter //

create procedure `usp\_return\_book` (in inBorrowID bigint,in inOperatorRet varchar(20))

begin

set @OverDay=DATEDIFF(CURRENT\_DATE,(select ldDateRetPlan from TB\_Borrow where BorrowID=inBorrowID));

if @OverDay < 0 then

set @OverDay = 0;

end if;

select PunishRate into @Rate from TB\_ReaderType where rdType=(

select TB\_Reader.rdType from TB\_Reader where TB\_Reader.rdID=(

select TB\_Borrow.rdID from TB\_Borrow where TB\_Borrow.BorrowID=inBorrowID

)

);

set @PunishMoney=@OverDay \* @Rate;

update TB\_Borrow set ldDateRetAct=CURRENT\_DATE, ldOverDay=@OverDay, ldOverMoney=@PunishMoney, ldPunishMoney=@PunishMoney, lsHasReturn=1, OperatorRet=inOperatorRet where BorrowID=inBorrowID;

update TB\_Book set bkStatus = '在馆' where bkID =(

select TB\_Borrow.bkID from TB\_Borrow where TB\_Borrow.BorrowID=inBorrowID

);

end //

delimiter ;

##### usp\_continue\_book

delimiter //

create procedure `usp\_continue\_book` (in inBorrowID bigint)

begin

select CanLendDay into @CanLendDay from TB\_ReaderType where rdType=(

select TB\_Reader.rdType from TB\_Reader where TB\_Reader.rdID=(

select TB\_Borrow.rdID from TB\_Borrow where TB\_Borrow.BorrowID=inBorrowID

)

);

set @ContinueTimes=(select ldContinueTimes from TB\_Borrow where BorrowID=inBorrowID)+1;

set @DateRetPlan=DATE\_ADD((select ldDateRetPlan from TB\_Borrow where BorrowID=inBorrowID), interval @CanLendDay day);

update TB\_Borrow set ldContinueTimes=@ContinueTimes, ldDateRetPlan=@DateRetPlan where BorrowID=inBorrowID;

end //

delimiter ;

##### usp\_get\_borrow\_count

delimiter //

create procedure `usp\_get\_borrow\_count` (in inrdID int,out outCount int)

begin

select count(\*) into outCount from TB\_Borrow where rdID=inrdID and lsHasReturn=0;

end //

delimiter ;

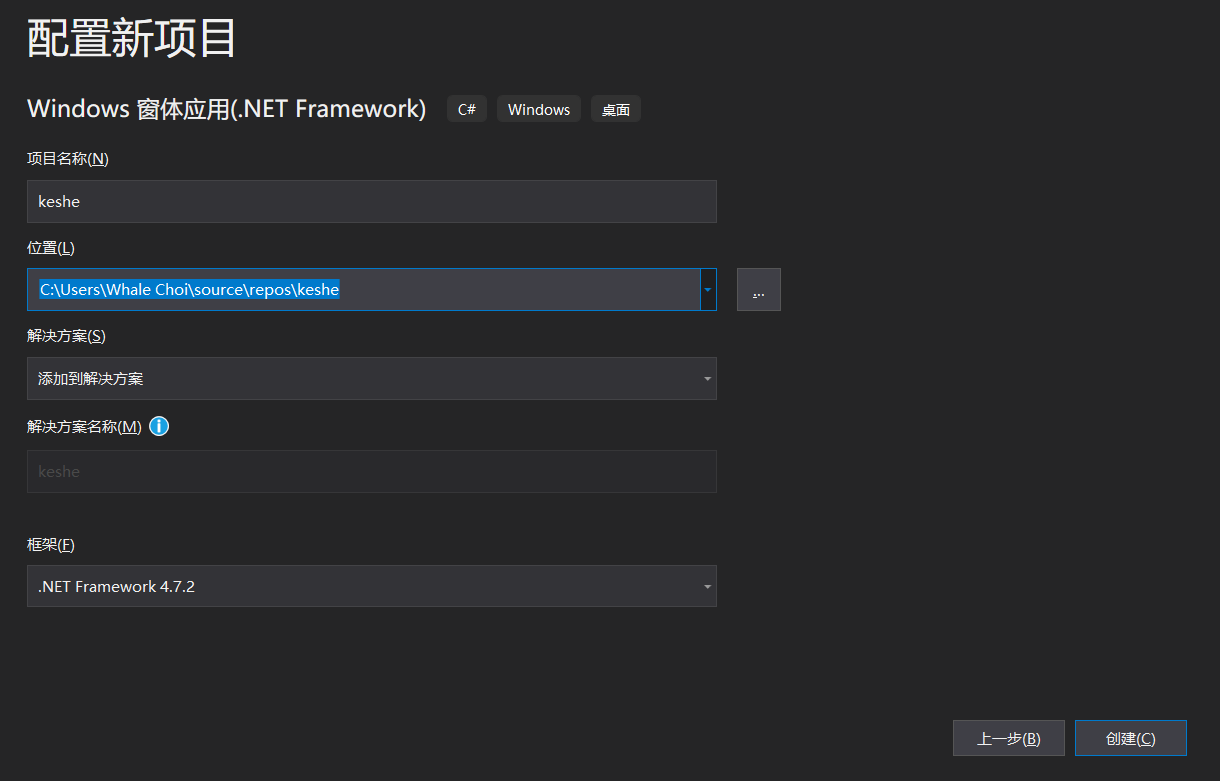
### 5.3 搭建三层架构解决方案

创建C#应用程序解决方案，包括4个项目：keshe（Windows窗口应用程序）、keshe.Model（类库）、keshe.DAL（类库）、keshe.BLL（类库）。

#### 1. 新建项目keshe（启动项目）

MS VS2019菜单：“文件”->“新建”->“项目”，在新建项目窗口中，选择“C#”->“Windows”->“Windows窗体应用(.NET Framework)”，选择项目文件的存储位置，并输入项目名称：keshe，将项目添加进解决方案，如图3。

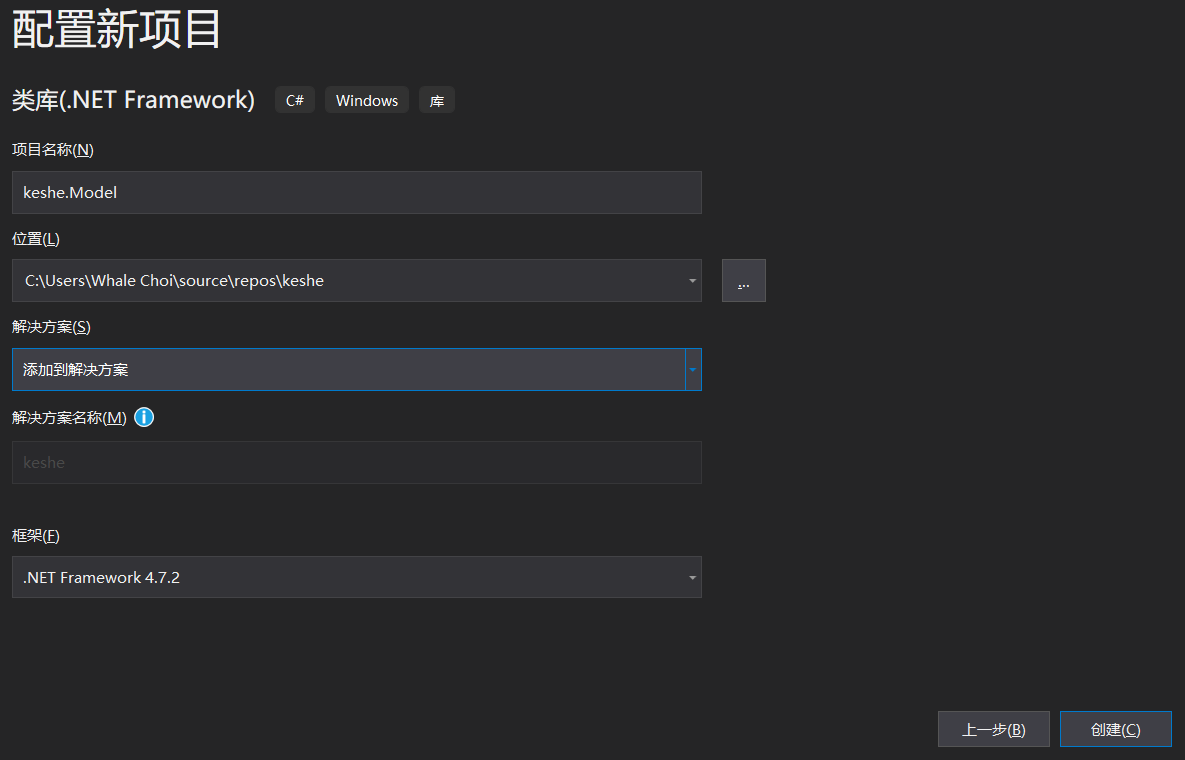
项目keshe即为UI层，该项目在VS .NET编译后产生keshe.exe文件。



**图3 新建项目keshe**

#### 2. 添加新项目keshe.Model

MS VS2019菜单：“文件”->“添加”->“新建项目”,选择“类库”，输入项目名称，如图4。VS .NET编译后产生keshe.Model.dll文件（.DLL后缀文件称为动态链接库）。



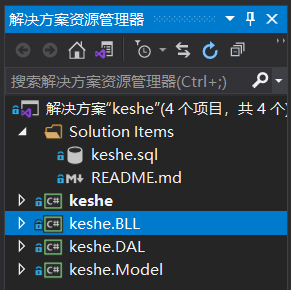
**图4 添加新建项目keshe.Model**

#### 3. 添加新项目keshe.DAL

操作与keshe.Model类似，项目名称设置为：keshe.DAL。VS .NET编译后产生keshe.DAL.dll文件。

#### 4. 添加新项目keshe.BLL

操作与keshe.Model类似，项目名称设置为：keshe.BLL。VS .NET编译后产生keshe.BLL.dll文件。至此，解决方案参见图5。



**图5 图书管理项目解决方案**

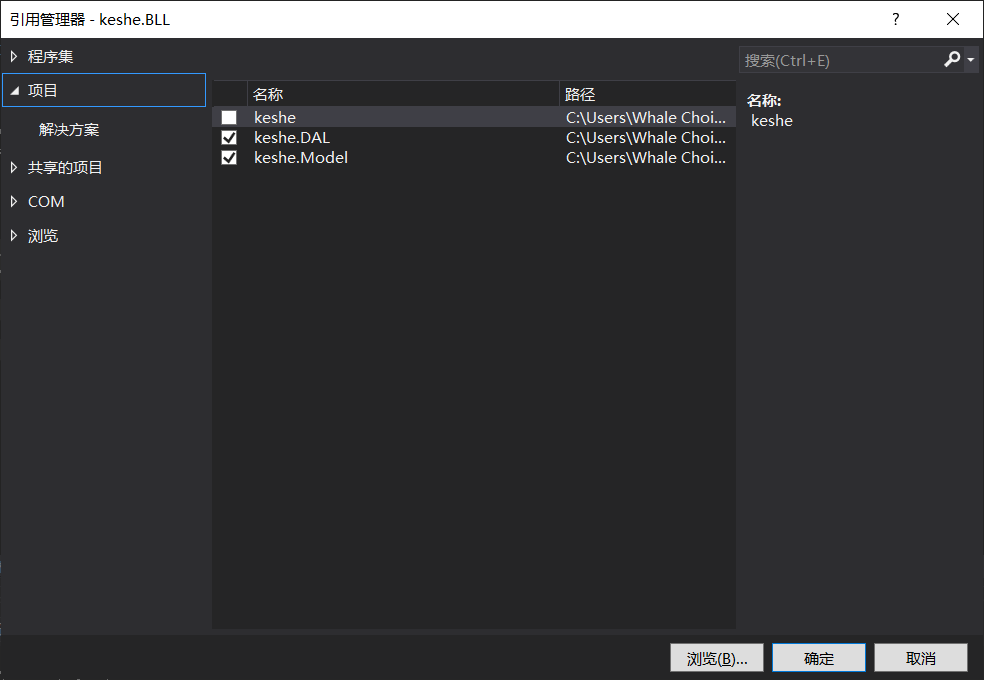
#### 5. 设置启动项目和项目引用关系

在解决方案中设置keshe为启动项目（“keshe”右键菜单“设为启动项目”）。

项目引用关系如表2所示。如keshe.BLL项目引用DAL和Model，其设置方法：解决方案中项目“keshe.BLL”的右键菜单“添加引用”，并按图6进行设置。

表2 项目引用关系。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | keshe | keshe.BLL | keshe.DAL | keshe.Model |
| keshe | **－** | **√** | **×** | **√** |
| keshe.BLL |  | **－** | **√** | **√** |
| keshe.DAL |  |  | **－** | **√** |
| keshe.Model |  |  |  | **－** |

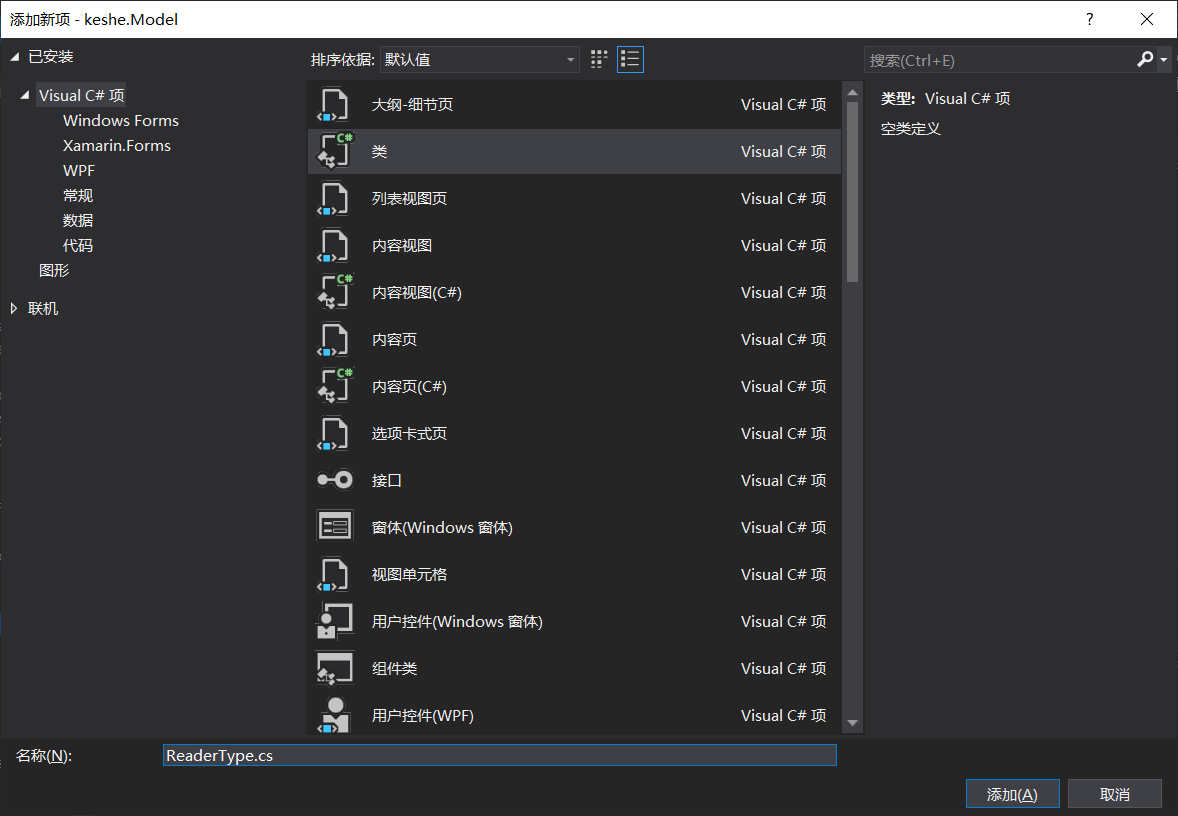


**图6 给keshe.BLL项目添加引用**

### 5.4 Model层

#### ReaderType类

在Model项目中添加新建项：C#代码类ReaderType.cs，如图7所示。



**图7 添加C#代码类文件ReaderType.cs**

表字段映射为实体类属性的基本方法如下：

private Int16 \_rdType = 0; /\* 读者类别 \*/

private string \_rdTypeName = ""; /\* 读者类别名称 \*/

private Int32 \_CanLendQty = 0; /\* 可借书数量 \*/

private Int32 \_CanLendDay = 0; /\* 可借书天数 \*/

private Int32 \_CanContinueTimes = 0; /\* 可续借的次数 \*/

private Single \_PunishRate = 0; /\* 罚款率（元/天） \*/

private Int16 \_DateValid = 0; /\* 证书有效期（年） \*/

对实体类属性进行封装（VS 2019可以Ctrl+R、Ctrl+E快捷封装）如下：

public short rdType { get => \_rdType; set => \_rdType = value; }

public string rdTypeName { get => \_rdTypeName; set => \_rdTypeName = value; }

public int CanLendQty { get => \_CanLendQty; set => \_CanLendQty = value; }

public int CanLendDay { get => \_CanLendDay; set => \_CanLendDay = value; }

public int CanContinueTimes { get => \_CanContinueTimes; set => \_CanContinueTimes = value; }

public float PunishRate { get => \_PunishRate; set => \_PunishRate = value; }

public short DateValid { get => \_DateValid; set => \_DateValid = value; }

#### Reader类

表字段映射为实体类属性的基本方法如下：

private Int32 \_rdID = 0; /\* 读者编号/借书证号 \*/

private string \_rdName = ""; /\* 读者姓名 \*/

private string \_rdSex = ""; /\* 性别：男/女 \*/

private Int16 \_rdType = 0; /\* 读者类别 \*/

private string \_rdDept = ""; /\* 单位代码/单位名称 \*/

private string \_rdPhone = ""; /\* 电话号码 \*/

private string \_rdEmail = ""; /\* 电子邮箱 \*/

private DateTime \_rdDateReg = default(DateTime); /\* 读者登记日期/办证日期 \*/

private byte[] \_rdPhoto = null; /\* 读者照片 \*/

private string \_rdStatus = ""; /\* 证件状态，3个：有效、挂失、注销 \*/

private Int32 \_rdBorrowQty = 0; /\* 已借书数量（缺省值0） \*/

private string \_rdPwd = ""; /\* 读者密码（初值123），32bit MD5加密存储 \*/

private Int16 \_rdAdminRoles = 0; /\* 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 \*/

对实体类属性进行封装（VS 2019可以Ctrl+R、Ctrl+E快捷封装）如下：

public int rdID { get => \_rdID; set => \_rdID = value; }

public string rdName { get => \_rdName; set => \_rdName = value; }

public string rdSex { get => \_rdSex; set => \_rdSex = value; }

public short rdType { get => \_rdType; set => \_rdType = value; }

public string rdDept { get => \_rdDept; set => \_rdDept = value; }

public string rdPhone { get => \_rdPhone; set => \_rdPhone = value; }

public string rdEmail { get => \_rdEmail; set => \_rdEmail = value; }

public DateTime rdDateReg { get => \_rdDateReg; set => \_rdDateReg = value; }

public byte[] rdPhoto { get => \_rdPhoto; set => \_rdPhoto = value; }

public string rdStatus { get => \_rdStatus; set => \_rdStatus = value; }

public int rdBorrowQty { get => \_rdBorrowQty; set => \_rdBorrowQty = value; }

public string rdPwd { get => \_rdPwd; set => \_rdPwd = value; }

public short rdAdminRoles { get => \_rdAdminRoles; set => \_rdAdminRoles = value; }

#### Borrow类

表字段映射为实体类属性的基本方法如下：

private Int64 \_BorrowID = 1; /\* 借书顺序号 \*/

private Int32 \_rdID = 0; /\* 读者序号 \*/

private Int32 \_bkID = 0; /\* 图书序号 \*/

private Int32 \_ldContinueTimes = 0; /\* 续借次数（第一次借时，记为0） \*/

private DateTime \_ldDateOut = default(DateTime); /\* 借书日期 \*/

private DateTime \_ldDateRetPlan = default(DateTime); /\* 应还日期 \*/

private DateTime \_ldDateRetAct = default(DateTime); /\* 实际还书日期 \*/

private Int32 \_ldOverDay = 0; /\* 超期天数 \*/

private decimal \_ldOverMoney = 0m; /\* 超期金额（应罚款金额） \*/

private decimal \_ldPunishMoney = 0m; /\* 罚款金额 \*/

private Boolean \_lsHasReturn = false; /\* 是否已经还书，缺省为0-未还 \*/

private string \_OperatorLend = ""; /\* 借书操作员 \*/

private string \_OperatorRet = ""; /\* 还书操作员 \*/

对实体类属性进行封装（VS 2019可以Ctrl+R、Ctrl+E快捷封装）如下：

public Int64 BorrowID { get => \_BorrowID; set => \_BorrowID = value; }

public int rdID { get => \_rdID; set => \_rdID = value; }

public int bkID { get => \_bkID; set => \_bkID = value; }

public int ldContinueTimes { get => \_ldContinueTimes; set => \_ldContinueTimes = value; }

public DateTime ldDateOut { get => \_ldDateOut; set => \_ldDateOut = value; }

public DateTime ldDateRetPlan { get => \_ldDateRetPlan; set => \_ldDateRetPlan = value; }

public DateTime ldDateRetAct { get => \_ldDateRetAct; set => \_ldDateRetAct = value; }

public int ldOverDay { get => \_ldOverDay; set => \_ldOverDay = value; }

public decimal ldOverMoney { get => \_ldOverMoney; set => \_ldOverMoney = value; }

public decimal ldPunishMoney { get => \_ldPunishMoney; set => \_ldPunishMoney = value; }

public bool lsHasReturn { get => \_lsHasReturn; set => \_lsHasReturn = value; }

public string OperatorLend { get => \_OperatorLend; set => \_OperatorLend = value; }

public string OperatorRet { get => \_OperatorRet; set => \_OperatorRet = value; }

#### Book类

表字段映射为实体类属性的基本方法如下：

private Int32 \_bkID = 0; /\* 图书序号 \*/

private string \_bkCode = ""; /\* 图书编号或条码号（前文中的书号） \*/

private string \_bkName = ""; /\* 书名 \*/

private string \_bkAuthor = ""; /\* 作者 \*/

private string \_bkPress = ""; /\* 出版社 \*/

private DateTime \_bkDatePress = default(DateTime); /\* 出版日期 \*/

private string \_bkISBN = ""; /\* ISBN书号 \*/

private string \_bkCatalog = ""; /\* 分类号（如：TP316-21/123） \*/

private Int16 \_bkLanguage = 0; /\* 语言：0-中文，1-英文，2-日文，3-俄文，4-德文，5-法文 \*/

private Int32 \_bkPages = 0; /\* 页数 \*/

private decimal \_bkPrice = 0m; /\* 价格 \*/

private DateTime \_bkDateIn = default(DateTime); /\* 入馆日期 \*/

private string \_bkBrief = ""; /\* 内容简介 \*/

private byte[] \_bkCover = null; /\* 图书封面照片 \*/

private string \_bkStatus = ""; /\* 图书状态：在馆、借出、遗失、变卖、销毁 \*/

对实体类属性进行封装（VS 2019可以Ctrl+R、Ctrl+E快捷封装）如下：

public int bkID { get => \_bkID; set => \_bkID = value; }

public string bkCode { get => \_bkCode; set => \_bkCode = value; }

public string bkName { get => \_bkName; set => \_bkName = value; }

public string bkAuthor { get => \_bkAuthor; set => \_bkAuthor = value; }

public string bkPress { get => \_bkPress; set => \_bkPress = value; }

public DateTime bkDatePress { get => \_bkDatePress; set => \_bkDatePress = value; }

public string bkISBN { get => \_bkISBN; set => \_bkISBN = value; }

public string bkCatalog { get => \_bkCatalog; set => \_bkCatalog = value; }

public short bkLanguage { get => \_bkLanguage; set => \_bkLanguage = value; }

public int bkPages { get => \_bkPages; set => \_bkPages = value; }

public decimal bkPrice { get => \_bkPrice; set => \_bkPrice = value; }

public DateTime bkDateIn { get => \_bkDateIn; set => \_bkDateIn = value; }

public string bkBrief { get => \_bkBrief; set => \_bkBrief = value; }

public byte[] bkCover { get => \_bkCover; set => \_bkCover = value; }

public string bkStatus { get => \_bkStatus; set => \_bkStatus = value; }

#### UserAction类

表字段映射为实体类属性的基本方法如下：

private string \_actionSource = "ReaderType"; // 来源 -> ReaderType Reader Book Borrow

private Object \_actionModel = null; // 需要进行操作的Model对象 -> Model(ReaderType、Reader、Book、Borrow)

private string \_actionType = "Add"; // 操作类型 -> Add Update Delete

private string \_actionDescription = ""; // 动作描述信息 -> 提供给用户的提示信息

对实体类属性进行封装（VS 2019可以Ctrl+R、Ctrl+E快捷封装）如下：

public string actionSource { get => \_actionSource; set => \_actionSource = value; }

public object actionModel { get => \_actionModel; set => \_actionModel = value; }

public string actionType { get => \_actionType; set => \_actionType = value; }

public string actionDescription { get => \_actionDescription; set => \_actionDescription = value; }

复制构造函数

public UserAction(UserAction r)

{

this.actionSource = r.actionSource;

this.actionModel = this.DeepCopy<object>(r.actionModel);

this.actionType = r.actionType;

this.actionDescription = r.actionDescription;

}

深复制模板函数

private T DeepCopy<T>(T obj)

{

object retval;

using (MemoryStream ms = new MemoryStream())

{

BinaryFormatter bf = new BinaryFormatter();

//序列化成流

bf.Serialize(ms, obj);

ms.Seek(0, SeekOrigin.Begin);

//反序列化成对象

retval = bf.Deserialize(ms);

ms.Close();

}

return (T)retval;

}

### 4.6 DAL层类

#### MySqlHelper类

在keshe.DAL项目中添加MySqlHelper.cs文件，该类是开源的MySqlHelper类（参考https://github.com/leomuller/MySqlHelper）。

MySqlHelper类主要包括的成员函数有：执行SQL语句、执行存储过程等。

在原版基础上此外还添加了DataTable转List<T>、DataRow转T、Null转DBNull。

##### DataTableToT

该方法用于将数据表转换成顺序链表，详细代码如下：

public static List<T> DataTableToT<T>(DataTable source) where T : class, new()

{

List<T> itemlist = null;

if (source == null || source.Rows.Count == 0)

{

return itemlist;

}

itemlist = new List<T>();

T item = null;

Type targettype = typeof(T);

Type ptype = null;

Object value = null;

foreach (DataRow dr in source.Rows)

{

item = new T();

foreach (PropertyInfo pi in targettype.GetProperties())

{

if (pi.CanWrite && source.Columns.Contains(pi.Name))

{

if (!(pi.PropertyType.FullName == "System.Byte[]" && dr[pi.Name] == DBNull.Value))

{

ptype = Type.GetType(pi.PropertyType.FullName);

value = Convert.ChangeType(dr[pi.Name], ptype);

pi.SetValue(item, value, null);

}

}

}

itemlist.Add(item);

}

return itemlist;

}

##### DataRowToT

该方法用于将数据行转换成实体类，详细代码如下：

public static T DataRowToT<T>(DataRow source) where T : class, new()

{

T item = null;

if (source == null)

{

return item;

}

item = new T();

Type targettype = typeof(T);

Type ptype = null;

Object value = null;

foreach (PropertyInfo pi in targettype.GetProperties())

{

if (pi.CanWrite && source.Table.Columns.Contains(pi.Name))

{

if (!(pi.PropertyType.FullName == "System.Byte[]" && source[pi.Name] == DBNull.Value))

{

ptype = Type.GetType(pi.PropertyType.FullName);

value = Convert.ChangeType(source[pi.Name], ptype);

pi.SetValue(item, value, null);

}

}

}

return item;

}

##### NullToMySqlNull

该方法用于数据库Null与C# Null的转换，详细代码如下：

public static object NullToMySqlNull(object obj)

{

if (obj == null)

{

return DBNull.Value;

}

return obj;

}

#### ReaderTypeDAL类

DAL层的类主要实现对应数据表的插、删、改、查等操作，以及存储过程的调用。

首先，在DAL项目中添加C#代码类文件-ReaderTypeDAL.cs，并在文件中添加引用：

using keshe.Model;

using MySql.Data.MySqlClient;

using System;

using System.Data;

并在类中添加一个常量

private static string \_strConnection = "server=localhost; uid=root; pwd=qweasdwsad; database=keshe";

然后，在ReaderTypeDAL类中添加对表TB\_ReaderType记录的插入Add()、删除Delete()、修改Update()等方法。

##### Add

该方法用于向数据库添加读者类型，详细代码如下：

public static int Add(ReaderType readertype)

{

int rows = 0;

string sql = "insert into TB\_ReaderType(rdType, rdTypeName, CanLendQty, CanLendDay, CanContinueTimes, PunishRate, DateValid)"

+ " values(?rdType, ?rdTypeName, ?CanLendQty, ?CanLendDay, ?CanContinueTimes, ?PunishRate, ?DateValid)";

MySqlParameter[] parameters =

{

new MySqlParameter("?rdType",MySqlDbType.Int16),

new MySqlParameter("?rdTypeName",MySqlDbType.VarChar,20),

new MySqlParameter("?CanLendQty",MySqlDbType.Int32),

new MySqlParameter("?CanLendDay",MySqlDbType.Int32),

new MySqlParameter("?CanContinueTimes",MySqlDbType.Int32),

new MySqlParameter("?PunishRate",MySqlDbType.Float),

new MySqlParameter("?DateValid",MySqlDbType.Int16)

};

parameters[0].Value = readertype.rdType;

parameters[1].Value = readertype.rdTypeName;

parameters[2].Value = readertype.CanLendQty;

parameters[3].Value = readertype.CanLendDay;

parameters[4].Value = readertype.CanContinueTimes;

parameters[5].Value = readertype.PunishRate;

parameters[6].Value = readertype.DateValid;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Delete

该方法用于删除读者类别，详细代码如下：

public static int Delete(ReaderType readertype)

{

int rows = 0;

string sql = "delete from TB\_ReaderType where rdType=?rdType";

MySqlParameter[] parameters = { new MySqlParameter("?rdType", MySqlDbType.Int16) };

parameters[0].Value = readertype.rdType;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Update

该方法用于更新读者类别，详细代码如下：

public static int Update(ReaderType readertype)

{

int rows = 0;

string sql = "update TB\_ReaderType set "

+ "rdTypeName=?rdTypeName, "

+ "CanLendQty=?CanLendQty, "

+ "CanLendDay=?CanLendDay, "

+ "CanContinueTimes=?CanContinueTimes, "

+ "PunishRate=?PunishRate, "

+ "DateValid=?DateValid "

+ "where rdType=?rdType";

MySqlParameter[] parameters =

{

new MySqlParameter("?rdTypeName",MySqlDbType.VarChar,20),

new MySqlParameter("?CanLendQty",MySqlDbType.Int32),

new MySqlParameter("?CanLendDay",MySqlDbType.Int32),

new MySqlParameter("?CanContinueTimes",MySqlDbType.Int32),

new MySqlParameter("?PunishRate",MySqlDbType.Float),

new MySqlParameter("?DateValid",MySqlDbType.Int16),

new MySqlParameter("?rdType",MySqlDbType.Int16)

};

parameters[0].Value = readertype.rdTypeName;

parameters[1].Value = readertype.CanLendQty;

parameters[2].Value = readertype.CanLendDay;

parameters[3].Value = readertype.CanContinueTimes;

parameters[4].Value = readertype.PunishRate;

parameters[5].Value = readertype.DateValid;

parameters[6].Value = readertype.rdType;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### GetDRByID

该方法使用读者类型id获取数据行，具体代码如下：

public static DataRow GetDRByID(Int16 rdType)

{

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

MySqlParameter[] parameters = { new MySqlParameter("?rdType", rdType) };

DataTable dt = null;

dt = MySqlHelper.GetData(\_strConnection, CommandType.Text, sql, parameters);

DataRow dr = null;

if (dt == null || dt.Rows.Count == 0)

return dr;

else

{

dr = dt.Rows[0];

return dr;

}

}

##### GetObjectByID

该方法使用读者类别id获取读者类别实体对象，具体代码如下：

public static ReaderType GetObjectByID(Int16 rdType)

{

DataRow dr = GetDRByID(rdType);

return MySqlHelper.DataRowToT<ReaderType>(dr);

}

##### GetCanLendQtyByID

该方法用于获取指定读者类别可借书天数，具体代码如下：

public static Int32 GetCanLendQtyByID(Int16 rdType)

{

DataRow dr = GetDRByID(rdType);

return MySqlHelper.DataRowToT<ReaderType>(dr).CanLendQty;

}

#### ReaderDAL类

DAL层的类主要实现对应数据表的插、删、改、查等操作，以及存储过程的调用。

首先，在DAL项目中添加C#代码类文件-ReaderDAL.cs，并在文件中添加引用：

using keshe.Model;

using MySql.Data.MySqlClient;

using System;

using System.Data;

并在类中添加一个常量

private static string \_strConnection = "server=localhost; uid=root; pwd=qweasdwsad; database=keshe";

然后，在ReaderDAL类中添加对表TB\_Reader记录的插入Add()、删除Delete()、修改Update()等方法。

##### Add

该方法为添加读者，具体代码如下：

public static int Add(Reader reader)

{

int rows = 0;

string sql = "insert into TB\_Reader(rdID, rdName, rdSex, rdType, rdDept, rdPhone, rdEmail, rdDateReg, rdPhoto, rdStatus, rdBorrowQty, rdPwd, rdAdminRoles)"

+ " values(?rdID, ?rdName, ?rdSex, ?rdType, ?rdDept, ?rdPhone, ?rdEmail, ?rdDateReg, ?rdPhoto, ?rdStatus, ?rdBorrowQty, ?rdPwd, ?rdAdminRoles)";

MySqlParameter[] parameters =

{

new MySqlParameter("?rdID",MySqlDbType.Int32),

new MySqlParameter("?rdName",MySqlDbType.VarChar,20),

new MySqlParameter("?rdSex",MySqlDbType.VarChar,2),

new MySqlParameter("?rdType",MySqlDbType.Int16),

new MySqlParameter("?rdDept",MySqlDbType.VarChar,20),

new MySqlParameter("?rdPhone",MySqlDbType.VarChar,25),

new MySqlParameter("?rdEmail",MySqlDbType.VarChar,25),

new MySqlParameter("?rdDateReg",MySqlDbType.DateTime),

new MySqlParameter("?rdPhoto",MySqlDbType.MediumBlob),

new MySqlParameter("?rdStatus",MySqlDbType.VarChar,2),

new MySqlParameter("?rdBorrowQty",MySqlDbType.Int32),

new MySqlParameter("?rdPwd",MySqlDbType.VarChar,32),

new MySqlParameter("?rdAdminRoles",MySqlDbType.Int16)

};

parameters[0].Value = reader.rdID;

parameters[1].Value = reader.rdName;

parameters[2].Value = reader.rdSex;

parameters[3].Value = reader.rdType;

parameters[4].Value = reader.rdDept;

parameters[5].Value = reader.rdPhone;

parameters[6].Value = reader.rdEmail;

parameters[7].Value = reader.rdDateReg;

parameters[8].Value = reader.rdPhoto;

parameters[9].Value = reader.rdStatus;

parameters[10].Value = reader.rdBorrowQty;

parameters[11].Value = reader.rdPwd;

parameters[12].Value = reader.rdAdminRoles;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Delete

该方法为删除读者，具体代码如下：

public static int Delete(Reader reader)

{

int rows = 0;

string sql = "delete from TB\_Reader where rdID=?rdID";

MySqlParameter[] parameters = { new MySqlParameter("?rdID", MySqlDbType.Int32) };

parameters[0].Value = reader.rdID;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Update

该方法为更新读者信息，具体代码如下：

public static int Update(Reader reader)

{

int rows = 0;

string sql = "update TB\_Reader set "

+ "rdName=?rdName, "

+ "rdSex=?rdSex, "

+ "rdType=?rdType, "

+ "rdDept=?rdDept, "

+ "rdPhone=?rdPhone, "

+ "rdEmail=?rdEmail, "

+ "rdDateReg=?rdDateReg, "

+ "rdPhoto=?rdPhoto, "

+ "rdStatus=?rdStatus, "

+ "rdBorrowQty=?rdBorrowQty, "

+ "rdPwd=?rdPwd, "

+ "rdAdminRoles=?rdAdminRoles "

+ "where rdID=?rdID";

MySqlParameter[] parameters =

{

new MySqlParameter("?rdName",MySqlDbType.VarChar,20),

new MySqlParameter("?rdSex",MySqlDbType.VarChar,2),

new MySqlParameter("?rdType",MySqlDbType.Int16),

new MySqlParameter("?rdDept",MySqlDbType.VarChar,20),

new MySqlParameter("?rdPhone",MySqlDbType.VarChar,25),

new MySqlParameter("?rdEmail",MySqlDbType.VarChar,25),

new MySqlParameter("?rdDateReg",MySqlDbType.DateTime),

new MySqlParameter("?rdPhoto",MySqlDbType.MediumBlob),

new MySqlParameter("?rdStatus",MySqlDbType.VarChar,2),

new MySqlParameter("?rdBorrowQty",MySqlDbType.Int32),

new MySqlParameter("?rdPwd",MySqlDbType.VarChar,32),

new MySqlParameter("?rdAdminRoles",MySqlDbType.Int16),

new MySqlParameter("?rdID",MySqlDbType.Int32)

};

parameters[0].Value = reader.rdName;

parameters[1].Value = reader.rdSex;

parameters[2].Value = reader.rdType;

parameters[3].Value = reader.rdDept;

parameters[4].Value = reader.rdPhone;

parameters[5].Value = reader.rdEmail;

parameters[6].Value = reader.rdDateReg;

parameters[7].Value = reader.rdPhoto;

parameters[8].Value = reader.rdStatus;

parameters[9].Value = reader.rdBorrowQty;

parameters[10].Value = reader.rdPwd;

parameters[11].Value = reader.rdAdminRoles;

parameters[12].Value = reader.rdID;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### GetDRByID

该方法使用读者id获取数据行，具体代码如下：

public static DataRow GetDRByID(Int32 rdID)

{

string sql = "select \* from TB\_Reader where rdID=?rdID";

MySqlParameter[] parameters = { new MySqlParameter("?rdID", rdID) };

DataTable dt = null;

dt = MySqlHelper.GetData(\_strConnection, CommandType.Text, sql, parameters);

DataRow dr = null;

if (dt == null || dt.Rows.Count == 0)

return dr;

else

{

dr = dt.Rows[0];

return dr;

}

}

##### GetObjectByID

该方法为使用读者id获取读者实体对象，具体代码如下：

public static Reader GetObjectByID(Int32 rdID)

{

DataRow dr = GetDRByID(rdID);

return MySqlHelper.DataRowToT<Reader>(dr);

}

##### Searchby

该方法为按条件模糊查询，第一个参数为查找方式（不限制，读者id，读者名称，性别，部门，电话，邮箱，状态）、第二个参数为查询的内容、第三个参数为从第几行返回数据、第四个参数为返回的查询条数，具体代码如下：

public static DataTable Searchby(string type, string content, int row, int number)

{

string sql = $"select rdID as 读者ID, rdName as 读者姓名, rdSex as 性别, rdPhone as 电话号码, rdStatus as 证件状态, rdAdminRoles as 职权 from TB\_Reader";

MySqlParameter parameter = null;

switch (type)

{

case "\_ALL":

sql = sql + $" limit {row}, {number}";

return MySqlHelper.GetData(\_strConnection, CommandType.Text, sql);

case "rdID":

sql = sql + $" where rdID=?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.Int32);

parameter.Value = Int32.Parse(content);

break;

case "rdName":

sql = sql + $" where rdName like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 20);

parameter.Value = $"%{content}%";

break;

case "rdSex":

sql = sql + $" where rdSex=?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 2);

parameter.Value = content;

break;

case "rdDept":

sql = sql + $" where rdDept like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 20);

parameter.Value = $"%{content}%";

break;

case "rdPhone":

sql = sql + $" where rdPhone like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 25);

parameter.Value = $"{content}%";

break;

case "rdEmail":

sql = sql + $" where rdEmail like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 25);

parameter.Value = $"{content}%";

break;

case "rdStatus":

sql = sql + $" where rdStatus=?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 2);

parameter.Value = content;

break;

default:

throw (new Exception("Error search type!"));

}

MySqlParameter[] parameters = { parameter };

return MySqlHelper.GetData(\_strConnection, CommandType.Text, sql, parameters);

}

##### Password

该方法为修改读者密码，需要提供被修改后的读者实体对象，具体代码如下：

public static int Password(Reader reader)

{

int rows = 0;

string sql = "update TB\_Reader set "

+ "rdPwd=?rdPwd "

+ "where rdID=?rdID";

MySqlParameter[] parameters =

{

new MySqlParameter("?rdPwd",MySqlDbType.VarChar,32),

new MySqlParameter("?rdID",MySqlDbType.Int32)

};

parameters[0].Value = reader.rdPwd;

parameters[1].Value = reader.rdID;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Permission

该方法为修改读者权限，需要提供修改后的读者对象，具体代码如下：

public static int Permission(Reader reader)

{

int rows = 0;

string sql = "update TB\_Reader set "

+ "rdAdminRoles=?rdAdminRoles "

+ "where rdID=?rdID";

MySqlParameter[] parameters =

{

new MySqlParameter("?rdAdminRoles",MySqlDbType.Int16),

new MySqlParameter("?rdID",MySqlDbType.Int32)

};

parameters[0].Value = reader.rdAdminRoles;

parameters[1].Value = reader.rdID;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

#### BorrowDAL类

DAL层的类主要实现对应数据表的插、删、改、查等操作，以及存储过程的调用。

首先，在DAL项目中添加C#代码类文件- BorrowDAL.cs，并在文件中添加引用：

using keshe.Model;

using MySql.Data.MySqlClient;

using System;

using System.Data;

并在类中添加一个常量

private static string \_strConnection = "server=localhost; uid=root; pwd=qweasdwsad; database=keshe";

然后，在BorrowDAL类中添加对表TB\_Borrow记录的插入Add()、删除Delete()、修改Update()等方法。

##### Borrow

该方法为借书方法，通过调用数据库的存储过程usp\_borrow\_book实现，具体代码如下：

public static int Borrow(Borrow borrow)

{

int rows = 0;

MySqlParameter[] parameters =

{

new MySqlParameter("?inrdID",MySqlDbType.Int32),

new MySqlParameter("?inbkID",MySqlDbType.Int32),

new MySqlParameter("?inOperatorLend",MySqlDbType.VarChar,20)

};

parameters[0].Value = borrow.rdID;

parameters[0].Direction = ParameterDirection.Input;

parameters[1].Value = borrow.bkID;

parameters[1].Direction = ParameterDirection.Input;

if (borrow.OperatorLend == null)

{

parameters[2].Value = "";

}

else

{

parameters[2].Value = borrow.OperatorLend;

}

parameters[2].Direction = ParameterDirection.Input;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.StoredProcedure, "usp\_borrow\_book", parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Delete

该方法用于删除借阅记录，具体代码如下：

public static int Delete(Borrow borrow)

{

int rows = 0;

string sql = "delete from TB\_Borrow where BorrowID=?BorrowID";

MySqlParameter[] parameters = { new MySqlParameter("?BorrowID", MySqlDbType.Int64) };

parameters[0].Value = borrow.BorrowID;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Return

该方法用于还书，通过调用数据库上的存储过程usp\_return\_book实现，具体代码如下：

public static int Return(Borrow borrow)

{

int rows = 0;

MySqlParameter[] parameters =

{

new MySqlParameter("?inBorrowID",MySqlDbType.Int64),

new MySqlParameter("?inOperatorRet",MySqlDbType.VarChar,20)

};

parameters[0].Value = borrow.BorrowID;

parameters[0].Direction = ParameterDirection.Input;

if (borrow.OperatorRet == null)

{

parameters[1].Value = "";

}

else

{

parameters[1].Value = borrow.OperatorRet;

}

parameters[1].Direction = ParameterDirection.Input;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.StoredProcedure, "usp\_return\_book", parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Update

该方法用于更新借阅记录，需要提供更新后的借阅实体对象，具体代码如下：

public static int Update(Borrow borrow)

{

int rows = 0;

string sql = "update TB\_Borrow set "

+ "rdID=?rdID, "

+ "bkID=?bkID, "

+ "ldContinueTimes=?ldContinueTimes, "

+ "ldDateOut=?ldDateOut, "

+ "ldDateRetPlan=?ldDateRetPlan, "

+ "ldDateRetAct=?ldDateRetAct, "

+ "ldOverDay=?ldOverDay, "

+ "ldOverMoney=?ldOverMoney, "

+ "ldPunishMoney=?ldPunishMoney, "

+ "lsHasReturn=?lsHasReturn, "

+ "OperatorLend=?OperatorLend, "

+ "OperatorRet=?OperatorRet "

+ "where BorrowID=?BorrowID";

MySqlParameter[] parameters =

{

new MySqlParameter("?rdID",MySqlDbType.Int32),

new MySqlParameter("?bkID",MySqlDbType.Int32),

new MySqlParameter("?ldContinueTimes",MySqlDbType.Int32),

new MySqlParameter("?ldDateOut",MySqlDbType.DateTime),

new MySqlParameter("?ldDateRetPlan",MySqlDbType.DateTime),

new MySqlParameter("?ldDateRetAct",MySqlDbType.DateTime),

new MySqlParameter("?ldOverDay",MySqlDbType.Int32),

new MySqlParameter("?ldOverMoney",MySqlDbType.NewDecimal),

new MySqlParameter("?ldPunishMoney",MySqlDbType.NewDecimal),

new MySqlParameter("?lsHasReturn",MySqlDbType.Bit),

new MySqlParameter("?OperatorLend",MySqlDbType.VarChar,20),

new MySqlParameter("?OperatorRet",MySqlDbType.VarChar,20),

new MySqlParameter("?BorrowID",MySqlDbType.NewDecimal)

};

parameters[0].Value = borrow.rdID;

parameters[1].Value = borrow.bkID;

parameters[2].Value = borrow.ldContinueTimes;

parameters[3].Value = borrow.ldDateOut;

parameters[4].Value = borrow.ldDateRetPlan;

parameters[5].Value = borrow.ldDateRetAct;

parameters[6].Value = borrow.ldOverDay;

parameters[7].Value = borrow.ldOverMoney;

parameters[8].Value = borrow.ldPunishMoney;

parameters[9].Value = borrow.lsHasReturn;

parameters[10].Value = borrow.OperatorLend;

parameters[11].Value = borrow.OperatorRet;

parameters[12].Value = borrow.BorrowID;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Continue

该方法用于续借，通过调用数据库上的存储过程usp\_continue\_book实现，具体代码如下：

public static int Continue(Borrow borrow)

{

int rows = 0;

MySqlParameter[] parameters =

{

new MySqlParameter("?inBorrowID",MySqlDbType.Int64)

};

parameters[0].Value = borrow.BorrowID;

parameters[0].Direction = ParameterDirection.Input;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.StoredProcedure, "usp\_continue\_book", parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Counter

该方法用于获取读者借了多少本书没还，通过调用数据库上的存储过程usp\_get\_borrow\_count，具体代码如下：

public static Int32 Counter(Int32 rdID)

{

Int32 counter = -1;

MySqlParameter[] parameters =

{

new MySqlParameter("?inrdID",MySqlDbType.Int32),

new MySqlParameter("?outCount",MySqlDbType.Int32)

};

parameters[0].Value = rdID;

parameters[0].Direction = ParameterDirection.Input;

parameters[1].Direction = ParameterDirection.Output;

try

{

MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.StoredProcedure, "usp\_get\_borrow\_count", parameters);

counter = (Int32)parameters[1].Value;

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return counter;

}

##### Searchby

该方法为模糊查询借阅记录，第一个参数为搜索方式（不限制、借阅id、图书id、借书操作员）、第二个参数为查询内容、第三个参数为从数据库的多少行开始查询、第四个参数为返回的查询结果条数，具体代码如下：

public static DataTable Searchby(string type, string content, int row, int number, int rdID)

{

string sql = $"select BorrowID as 借书顺序号, bkID as 图书序号, ldContinueTimes as 续借次数, ldDateOut as 借书日期, ldDateRetPlan as 应还日期, lsHasReturn as 是否还书, OperatorLend as 借书操作员 from TB\_Borrow where rdID={rdID}";

MySqlParameter parameter = null;

switch (type)

{

case "\_ALL":

sql = sql + $" limit {row}, {number}";

return MySqlHelper.GetData(\_strConnection, CommandType.Text, sql);

case "BorrowID":

sql = sql + $" and BorrowID=?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.Int64);

parameter.Value = Int64.Parse(content);

break;

case "bkID":

sql = sql + $" and bkID=?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.Int32);

parameter.Value = Int32.Parse(content);

break;

case "OperatorLend":

sql = sql + $" and OperatorLend like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 20);

parameter.Value = $"%{content}%";

break;

default:

throw (new Exception("Error search type!"));

}

MySqlParameter[] parameters = { parameter };

return MySqlHelper.GetData(\_strConnection, CommandType.Text, sql, parameters);

}

#### BookDAL类

DAL层的类主要实现对应数据表的插、删、改、查等操作，以及存储过程的调用。

首先，在DAL项目中添加C#代码类文件- BookDAL.cs，并在文件中添加引用：

using keshe.Model;

using MySql.Data.MySqlClient;

using System;

using System.Data;

并在类中添加一个常量

private static string \_strConnection = "server=localhost; uid=root; pwd=qweasdwsad; database=keshe";

然后，在BookDAL类中添加对表TB\_ Book记录的插入Add()、删除Delete()、修改Update()等方法。

##### Add

该方法为添加图书，具体代码如下：

public static int Add(Book book)

{

int rows = 0;

string sql = "insert into TB\_Book(bkID, bkCode, bkName, bkAuthor, bkPress, bkDatePress, bkISBN, bkCatalog, bkLanguage, bkPages, bkPrice, bkDateIn, bkBrief, bkCover, bkStatus)"

+ " values(?bkID, ?bkCode, ?bkName, ?bkAuthor, ?bkPress, ?bkDatePress, ?bkISBN, ?bkCatalog, ?bkLanguage, ?bkPages, ?bkPrice, ?bkDateIn, ?bkBrief, ?bkCover, ?bkStatus)";

MySqlParameter[] parameters =

{

new MySqlParameter("?bkID",MySqlDbType.Int32),

new MySqlParameter("?bkCode",MySqlDbType.VarChar,20),

new MySqlParameter("?bkName",MySqlDbType.VarChar,50),

new MySqlParameter("?bkAuthor",MySqlDbType.VarChar,30),

new MySqlParameter("?bkPress",MySqlDbType.VarChar,50),

new MySqlParameter("?bkDatePress",MySqlDbType.DateTime),

new MySqlParameter("?bkISBN",MySqlDbType.VarChar,15),

new MySqlParameter("?bkCatalog",MySqlDbType.VarChar,30),

new MySqlParameter("?bkLanguage",MySqlDbType.Int16),

new MySqlParameter("?bkPages",MySqlDbType.Int32),

new MySqlParameter("?bkPrice",MySqlDbType.NewDecimal),

new MySqlParameter("?bkDateIn",MySqlDbType.DateTime),

new MySqlParameter("?bkBrief",MySqlDbType.Text),

new MySqlParameter("?bkCover",MySqlDbType.MediumBlob),

new MySqlParameter("?bkStatus",MySqlDbType.VarChar,2)

};

parameters[0].Value = book.bkID;

parameters[1].Value = book.bkCode;

parameters[2].Value = book.bkName;

parameters[3].Value = book.bkAuthor;

parameters[4].Value = book.bkPress;

parameters[5].Value = book.bkDatePress;

parameters[6].Value = book.bkISBN;

parameters[7].Value = book.bkCatalog;

parameters[8].Value = book.bkLanguage;

parameters[9].Value = book.bkPages;

parameters[10].Value = book.bkPrice;

parameters[11].Value = book.bkDateIn;

parameters[12].Value = book.bkBrief;

parameters[13].Value = book.bkCover;

parameters[14].Value = book.bkStatus;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Delete

该方法为删除图书，具体代码如下：

public static int Delete(Book book)

{

int rows = 0;

string sql = "delete from TB\_Book where bkID=?bkID";

MySqlParameter[] parameters = { new MySqlParameter("?bkID", MySqlDbType.Int32) };

parameters[0].Value = book.bkID;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### Update

该方法为更新图书信息，具体代码如下：

public static int Update(Book book)

{

int rows = 0;

string sql = "update TB\_Book set "

+ "bkCode=?bkCode, "

+ "bkName=?bkName, "

+ "bkAuthor=?bkAuthor, "

+ "bkPress=?bkPress, "

+ "bkDatePress=?bkDatePress, "

+ "bkISBN=?bkISBN, "

+ "bkCatalog=?bkCatalog, "

+ "bkLanguage=?bkLanguage, "

+ "bkPages=?bkPages, "

+ "bkPrice=?bkPrice, "

+ "bkDateIn=?bkDateIn, "

+ "bkBrief=?bkBrief, "

+ "bkCover=?bkCover, "

+ "bkStatus=?bkStatus "

+ "where bkID=?bkID";

MySqlParameter[] parameters =

{

new MySqlParameter("?bkCode",MySqlDbType.VarChar,20),

new MySqlParameter("?bkName",MySqlDbType.VarChar,50),

new MySqlParameter("?bkAuthor",MySqlDbType.VarChar,30),

new MySqlParameter("?bkPress",MySqlDbType.VarChar,50),

new MySqlParameter("?bkDatePress",MySqlDbType.DateTime),

new MySqlParameter("?bkISBN",MySqlDbType.VarChar,15),

new MySqlParameter("?bkCatalog",MySqlDbType.VarChar,30),

new MySqlParameter("?bkLanguage",MySqlDbType.Int16),

new MySqlParameter("?bkPages",MySqlDbType.Int32),

new MySqlParameter("?bkPrice",MySqlDbType.NewDecimal),

new MySqlParameter("?bkDateIn",MySqlDbType.DateTime),

new MySqlParameter("?bkBrief",MySqlDbType.Text),

new MySqlParameter("?bkCover",MySqlDbType.MediumBlob),

new MySqlParameter("?bkStatus",MySqlDbType.VarChar,2),

new MySqlParameter("?bkID",MySqlDbType.Int32)

};

parameters[0].Value = book.bkCode;

parameters[1].Value = book.bkName;

parameters[2].Value = book.bkAuthor;

parameters[3].Value = book.bkPress;

parameters[4].Value = book.bkDatePress;

parameters[5].Value = book.bkISBN;

parameters[6].Value = book.bkCatalog;

parameters[7].Value = book.bkLanguage;

parameters[8].Value = book.bkPages;

parameters[9].Value = book.bkPrice;

parameters[10].Value = book.bkDateIn;

parameters[11].Value = book.bkBrief;

parameters[12].Value = book.bkCover;

parameters[13].Value = book.bkStatus;

parameters[14].Value = book.bkID;

try

{

rows = MySqlHelper.ExecuteNonQuery(\_strConnection, CommandType.Text, sql, parameters);

}

catch (MySqlException ex)

{

throw new Exception(ex.Message);

}

return rows;

}

##### GetDRByID

该方法使用图书id获取数据行，具体代码如下：

public static DataRow GetDRByID(Int32 bkID)

{

string sql = "select \* from TB\_Book where bkID=?bkID";

MySqlParameter[] parameters = { new MySqlParameter("?bkID", bkID) };

DataTable dt = null;

dt = MySqlHelper.GetData(\_strConnection, CommandType.Text, sql, parameters);

DataRow dr = null;

if (dt == null || dt.Rows.Count == 0)

return dr;

else

{

dr = dt.Rows[0];

return dr;

}

}

##### GetObjectByID

该方法使用图书id获取图书实体对象，具体代码如下：

public static Book GetObjectByID(int bkID)

{

DataRow dr = GetDRByID(bkID);

return MySqlHelper.DataRowToT<Book>(dr);

}

##### GetLastObject

该方法获取数据库图书id最大的图书实体对象，具体代码如下：

public static Book GetLastObject()

{

string sql = "select \* from TB\_Book order by bkID DESC limit 1";

DataTable dt = null;

dt = MySqlHelper.GetData(\_strConnection, CommandType.Text, sql);

DataRow dr = null;

if (dt == null || dt.Rows.Count == 0)

return null;

else

{

dr = dt.Rows[0];

return MySqlHelper.DataRowToT<Book>(dr);

}

}

##### Searchby

该方法用于模糊查询，第一个参数是查询方式（不限、图书id、图书编号、书名、作者、出版社、ISBN）、第二个参数是查询内容、第三个参数是从数据的的多少行开始查询、第四个参数是返回的查询记录数，具体代码如下：

public static DataTable Searchby(string type, string content, int row, int number)

{

string sql = "select bkID as 图书序号, bkCode as 图书编号, bkName as 图书名称, bkAuthor as 图书作者, bkPress as 出版社名, bkISBN as 标准ISBN, bkDatePress as 出版日期, bkStatus as 图书状态 from TB\_Book";

MySqlParameter parameter = null;

switch (type)

{

case "\_ALL":

sql = sql + $" limit {row}, {number}";

return MySqlHelper.GetData(\_strConnection, CommandType.Text, sql);

case "bkID":

sql = sql + $" where bkID=?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.Int32);

parameter.Value = Int32.Parse(content);

break;

case "bkCode":

sql = sql + $" where bkCode like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 20);

parameter.Value = $"{content}%";

break;

case "bkName":

sql = sql + $" where bkName like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 50);

parameter.Value = $"%{content}%";

break;

case "bkAuthor":

sql = sql + $" where bkAuthor like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 30);

parameter.Value = $"%{content}%";

break;

case "bkPress":

sql = sql + $" where bkPress like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 50);

parameter.Value = $"%{content}%";

break;

case "bkISBN":

sql = sql + $" where bkISBN like ?content limit {row}, {number}";

parameter = new MySqlParameter("?content", MySqlDbType.VarChar, 15);

parameter.Value = $"%{content}%";

break;

default:

throw (new Exception("Error search type!"));

}

MySqlParameter[] parameters = { parameter };

return MySqlHelper.GetData(\_strConnection, CommandType.Text, sql, parameters);

}

### 4.7 BLL层类设计

#### loginControler类

此类为登录窗口的控制类，主要提供一些方法与数据库进行通信。

##### Login

该方法用于登录认证，参数为读者id、密码，返回登录结果，具体代码如下：

public static Int32 Login(Int32 rdID, string rdPwd)

{

Reader reader = ReaderDAL.GetObjectByID(rdID);

if (reader == null || rdID == 0)

return 1; // 1 -> 用户不存在；

if (reader.rdPwd != rdPwd)

return 2; // 2 -> 用户密码错误；

return 0; // 0 -> 登录成功；

}

##### GetReader

该方法用于使用读者id获取读者实体对象，具体代码如下：

public static Reader GetReader(Int32 rdID)

{

return ReaderDAL.GetObjectByID(rdID);

}

#### mainControler类

此类为主窗口的控制类，主要用于处理用户操作。

##### GetBorrowMax

该方法为使用读者获取该读者最多借书的本书，具体代码如下：

public static Int32 GetBorrowMax(Reader reader)

{

return ReaderTypeDAL.GetCanLendQtyByID(reader.rdType);

}

##### GetBorrowNotReturn

该方法为使用读者获取该读者没还书的本书，具体代码如下：

public static Int32 GetBorrowNotReturn(Reader reader)

{

return BorrowDAL.Counter(reader.rdID);

}

##### SubmitAction

该方法为提交用户操作，具体代码如下：

public static Int32 SubmitAction(UserAction action)

{

switch (action.actionSource)

{

case "ReaderType":

switch (action.actionType)

{

case "Add":

try

{

return ReaderTypeDAL.Add((ReaderType)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Update":

try

{

return ReaderTypeDAL.Update((ReaderType)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Delete":

try

{

return ReaderTypeDAL.Delete((ReaderType)action.actionModel);

}

catch (Exception)

{

return 0;

}

default:

throw new Exception("Error actionType!");

}

case "Reader":

switch (action.actionType)

{

case "Add":

try

{

return ReaderDAL.Add((Reader)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Update":

try

{

return ReaderDAL.Update((Reader)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Delete":

try

{

return ReaderDAL.Delete((Reader)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Password":

try

{

return ReaderDAL.Password((Reader)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Permission":

try

{

return ReaderDAL.Permission((Reader)action.actionModel);

}

catch (Exception)

{

return 0;

}

default:

throw new Exception("Error actionType!");

}

case "Book":

switch (action.actionType)

{

case "Add":

try

{

return BookDAL.Add((Book)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Update":

try

{

return BookDAL.Update((Book)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Delete":

try

{

return BookDAL.Delete((Book)action.actionModel);

}

catch (Exception)

{

return 0;

}

default:

throw new Exception("Error actionType!");

}

case "Borrow":

switch (action.actionType)

{

case "Borrow":

try

{

return BorrowDAL.Borrow((Borrow)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Update":

try

{

return BorrowDAL.Update((Borrow)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Delete":

try

{

return BorrowDAL.Delete((Borrow)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Return":

try

{

return BorrowDAL.Return((Borrow)action.actionModel);

}

catch (Exception)

{

return 0;

}

case "Continue":

try

{

return BorrowDAL.Continue((Borrow)action.actionModel);

}

catch (Exception)

{

return 0;

}

default:

throw new Exception("Error actionType!");

}

default:

throw (new Exception("Error actionSource!"));

}

}

#### bookAddControler类

改类为添加图书窗口的控制类，主要提供一些访问数据库的方法以及实用方法。

##### GetLastBkID

获取数据库最大的图书id，具体代码如下：

public static Int32 GetLastBkID()

{

Book book = BookDAL.GetLastObject();

if (book == null)

{

return -1;

}

return book.bkID;

}

##### ImageToByte

该方法用于将图片转换成字节数组，具体代码如下：

public static byte[] ImageToByte(Image image)

{

ImageFormat format = image.RawFormat;

using (MemoryStream ms = new MemoryStream())

{

if (format.Equals(ImageFormat.Jpeg))

{

image.Save(ms, ImageFormat.Jpeg);

}

else if (format.Equals(ImageFormat.Png))

{

image.Save(ms, ImageFormat.Png);

}

else if (format.Equals(ImageFormat.Bmp))

{

image.Save(ms, ImageFormat.Bmp);

}

byte[] buffer = new byte[ms.Length];

//Image.Save()会改变MemoryStream的Position，需要重新Seek到Begin

ms.Seek(0, SeekOrigin.Begin);

ms.Read(buffer, 0, buffer.Length);

return buffer;

}

}

##### ByteToImage

该方法用于将字节数组转换为图片，具体代码如下：

public static Image ByteToImage(byte[] buffer)

{

MemoryStream ms = new MemoryStream(buffer);

Image image = System.Drawing.Image.FromStream(ms);

return image;

}

#### bookDetailControler类

改类为修改图书窗口的控制类，主要提供一些访问数据库的方法以及实用方法。

##### GetBookbybkID

该方法用于通过图书id获取图书实体对象，具体代码如下：

public static Book GetBookbybkID(Int32 bkID)

{

return BookDAL.GetObjectByID(bkID);

}

#### bookSearchControler类

改类为查找图书窗口的控制类，主要提供一些访问数据库的方法以及实用方法。

##### GetDTby

该方法为搜索方法，同上，具体代码如下：

public static DataTable GetDTby(string type, string content, int row, int number)

{

return BookDAL.Searchby(type, content, row, number);

}

#### borrowSearchControler类

改类为查找借阅窗口的控制类，主要提供一些访问数据库的方法以及实用方法。

##### GetDTby

该方法为搜索方法，同上，具体代码如下：

public static DataTable GetDTby(string type, string content, int row, int number, int rdID)

{

return BorrowDAL.Searchby(type, content, row, number, rdID);

}

#### readerSearchControler类

改类为查找读者窗口的控制类，主要提供一些访问数据库的方法以及实用方法。

##### GetDTby

该方法为搜索方法，同上，具体代码如下：

public static DataTable GetDTby(string type, string content, int row, int number)

{

return ReaderDAL.Searchby(type, content, row, number);

}

### 4.8 UI层类设计

#### 单例模式

单例模式，属于创建类型的一种常用的[软件设计模式](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F/2117635)。通过单例模式的方法创建的类在当前进程中只有一个[实例](https://baike.baidu.com/item/%E5%AE%9E%E4%BE%8B/3794138)（根据需要，也有可能一个线程中属于单例，如：仅线程上下文内使用同一个实例）。

在该系统中，所有的窗体全部使用单例模式实现。

##### 具体实现案例

private static login \_instance = null; // 单例模式

public static login CreateInstance()

{

if (\_instance == null)

{

\_instance = new login();

}

return \_instance;

}

private login()

{

InitializeComponent();

}

public static bool isExist()

{

if (\_instance != null)

{

return true;

}

return false;

}

public static void disposeAll()

{

if (\_instance != null)

{

\_instance.Dispose();

\_instance = null;

bkID = -1;

}

}

#### Program(程序入口)

##### Main

进入死循环，登录窗口和主窗口循环显示，除非用户直接关闭主窗口或者点了登录界面的退出按钮才结束程序，具体代码如下：

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

while (true)

{

if (GlobalObject.reader == null)

{

Form \_login = login.CreateInstance();

\_login.ShowDialog();

}

else

{

Form \_main = main.CreateInstance();

\_main.ShowDialog();

}

}

}

#### GlobalObject(作用于整个UI层)

##### 属性

public static Reader reader = null; // 用户

public static bool borrowCardAdmin = false;

public static bool bookAdmin = false;

public static bool borrowAdmin = false;

public static bool systemAdmin = false;

public static readonly UserAction actionSource = new UserAction(); // 程序所有的用户行为由此复制

public static readonly ReaderType readerTypeSource = new ReaderType(); // actionSource 的 actionModel 为 ReaderType 时

public static readonly Reader readerSource = new Reader(); // actionSource 的 actionModel 为 Reader 时

public static readonly Book bookSource = new Book(); // actionSource 的 actionModel 为 Book 时

public static readonly Borrow borrowSource = new Borrow(); // actionSource 的 actionModel 为 Borrow 时

##### adminCheck

该方法用于检查管理员身份，具体代码如下：

private static void adminCheck()

{

if (GlobalObject.reader == null)

{

return;

}

if ((GlobalObject.reader.rdAdminRoles & 0b1000) == 0b1000)

{

GlobalObject.borrowCardAdmin = true;

}

if ((GlobalObject.reader.rdAdminRoles & 0b0100) == 0b0100)

{

GlobalObject.bookAdmin = true;

}

if ((GlobalObject.reader.rdAdminRoles & 0b0010) == 0b0010)

{

GlobalObject.borrowAdmin = true;

}

if ((GlobalObject.reader.rdAdminRoles & 0b0001) == 0b0001)

{

GlobalObject.systemAdmin = true;

}

}

##### Login

该方法用于登录，具体代码如下：

public static void login(Reader x)

{

if (x == null)

{

return;

}

GlobalObject.reader = x;

GlobalObject.adminCheck();

}

##### Logout

该方法用于登出，具体代码如下：

public static void logout()

{

GlobalObject.reader = null;

GlobalObject.borrowCardAdmin = false;

GlobalObject.bookAdmin = false;

GlobalObject.borrowAdmin = false;

GlobalObject.systemAdmin = false;

}

#### login

对应4.8.1 用户登录窗口

##### button\_login\_Click

按下登录按钮

private void button\_login\_Click(object sender, EventArgs e)

{

Int32 loginStatus = -1;

if (textBox\_rdID.Text == "")

{

MessageBox.Show("用户编号不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_rdPwd.Text == "")

{

MessageBox.Show("用户密码不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

try

{

loginStatus = loginControler.Login(Int32.Parse(textBox\_rdID.Text), EncryptProvider.Md5(textBox\_rdPwd.Text));

}

catch (Exception) { } // 捕捉到异常什么也不做

switch (loginStatus)

{

case 1:

MessageBox.Show("查无此用户！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

break;

case 2:

MessageBox.Show("密码错误！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

break;

case 0:

this.Visible = false;

GlobalObject.login(loginControler.GetReader(Int32.Parse(textBox\_rdID.Text)));

if (GlobalObject.reader.rdStatus == "挂失")

{

MessageBox.Show("您当前的借书证已被挂失，欲了解详细情况请与借书证管理员联系！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

GlobalObject.logout();

this.Visible = true;

break;

}

if (GlobalObject.reader.rdStatus == "注销")

{

MessageBox.Show("您当前的借书证已被注销，您不能登入此图书管理系统！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

GlobalObject.logout();

this.Visible = true;

break;

}

MessageBox.Show($"登录成功，{GlobalObject.reader.rdName}，欢迎使用 keshe 图书管理系统!", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

\_instance = null;

this.Dispose();

break;

case -1:

MessageBox.Show("登录失败，请检查您的网络连接！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

break;

default:

MessageBox.Show("登录失败，未知错误！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

break;

}

}

##### button\_exit\_Click

按下退出按钮

private void button\_exit\_Click(object sender, EventArgs e)

{

\_instance = null;

this.Dispose();

System.Environment.Exit(0);

}

##### textBox\_rdID\_KeyPress

在用户编号文本框中按下键盘按键，用于检测输入的值是否为数字

private void textBox\_rdID\_KeyPress(object sender, KeyPressEventArgs e)

{

// 用户编号只能输入数字！ (char)8表示退格键

if (!(Char.IsNumber(e.KeyChar)) && e.KeyChar != (char)8)

{

e.Handled = true;

}

else

{

e.Handled = false;

}

}

#### main

对应4.8.2 主界面窗口

##### 属性

public static Int32 BorrowMax = -1;

public static Int32 BorrowNotReturn = -1;

public static readonly BindingList<UserAction> ActionList = new BindingList<UserAction>();

private int index = -1;

##### addAction

该方法用于向ActionList添加用户操作，具体代码如下：

public static void addAction(UserAction action)

{

ActionList.Add(new UserAction(action)); // 添加拷贝后的用户操作，而不是操作本身的引用

}

##### WndProc

该方法为WndProc方法的重写，主要用于判断是代码致使窗口关闭还是用户点击关闭按钮，具体代码如下：

protected override void WndProc(ref Message msg)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_CLOSE = 0xF060;

if (msg.Msg == WM\_SYSCOMMAND && ((int)msg.WParam == SC\_CLOSE))

{

/// <summary>

/// 若用户点了关闭按钮，则执行下面代码

/// 调用Close()或者Dispose()方法不会触发

/// </summary>

if (canClose())

{

bookAdd.disposeAll();

bookSearch.disposeAll();

bookSearch\_reader.disposeAll();

readerSearch.disposeAll();

ActionList.Clear();

GlobalObject.logout();

BorrowMax = -1;

BorrowNotReturn = -1;

\_instance = null;

this.Dispose();

System.Environment.Exit(0);

}

return;

}

base.WndProc(ref msg);

}

##### dvg\_style

该方法用于设置datagridview的样式，具体代码如下：

private void dvg\_style()

{

dgv\_Actions.Columns["actionSource"].HeaderText = "操作源";

dgv\_Actions.Columns["actionModel"].Visible = false;

dgv\_Actions.Columns["actionType"].HeaderText = "操作类型";

dgv\_Actions.Columns["actionDescription"].HeaderText = "操作描述";

dgv\_Actions.Columns["actionSource"].FillWeight = 15;

dgv\_Actions.Columns["actionType"].FillWeight = 15;

dgv\_Actions.Columns["actionDescription"].FillWeight = 70;

}

##### canClose

该方法用于判断用户在关闭主窗口时是否还有其他操作，用于提示用户，具体代码如下：

private bool canClose()

{

if (bookAdd.isExist())

{

DialogResult dr = MessageBox.Show($"检测到您正在添加图书，是否继续？", "提示：", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr != DialogResult.OK)

{

return false;

}

}

if (bookSearch.isExist())

{

DialogResult dr = MessageBox.Show($"检测到您正在维护图书，是否继续？", "提示：", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr != DialogResult.OK)

{

return false;

}

}

if (bookSearch\_reader.isExist())

{

DialogResult dr = MessageBox.Show($"检测到您正在借阅图书，是否继续？", "提示：", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr != DialogResult.OK)

{

return false;

}

}

if (borrowSearch.isExist())

{

DialogResult dr = MessageBox.Show($"检测到您正在续借或归还图书，是否继续？", "提示：", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr != DialogResult.OK)

{

return false;

}

}

if (readerSearch.isExist())

{

DialogResult dr = MessageBox.Show($"检测到您正在进行权限管理，是否继续？", "提示：", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr != DialogResult.OK)

{

return false;

}

}

if (ActionList.Count != 0)

{

DialogResult dr = MessageBox.Show($"您还有 {ActionList.Count} 个操作未提交，是否继续？", "提示：", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr != DialogResult.OK)

{

return false;

}

}

return true;

}

##### main\_Load

主窗口加载时执行

private void main\_Load(object sender, EventArgs e)

{

BorrowMax = mainControler.GetBorrowMax(GlobalObject.reader);

BorrowNotReturn = mainControler.GetBorrowNotReturn(GlobalObject.reader);

if (BorrowNotReturn == -1)

{

MessageBox.Show("程序发生内部错误，即将强制退出！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

System.Environment.Exit(0);

}

this.Text = $"keshe 图书管理系统 v{Assembly.GetExecutingAssembly().GetName().Version.ToString()} 祝您阅读愉快！ 当前登录：{GlobalObject.reader.rdName}";

dgv\_Actions.DataSource = ActionList;

dvg\_style();

if (GlobalObject.borrowCardAdmin)

{

借书证管理ToolStripMenuItem.Visible = true;

}

if (GlobalObject.bookAdmin)

{

图书管理ToolStripMenuItem.Visible = true;

}

if (GlobalObject.borrowAdmin)

{

借阅管理ToolStripMenuItem.Visible = true;

}

if (GlobalObject.systemAdmin)

{

系统管理ToolStripMenuItem.Visible = true;

}

toolStripStatusLabel.Text = $"[Info] 加载完毕，就绪";

}

##### 关于ToolStripMenuItem\_Click

关于按钮被点击

private void 关于ToolStripMenuItem\_Click(object sender, EventArgs e)

{

AboutBox about = new AboutBox();

about.ShowDialog();

}

##### 退出系统ToolStripMenuItem\_Click

退出按钮被点击

private void 退出系统ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (canClose())

{

bookAdd.disposeAll();

bookSearch.disposeAll();

bookSearch\_reader.disposeAll();

readerSearch.disposeAll();

ActionList.Clear();

GlobalObject.logout();

BorrowMax = -1;

BorrowNotReturn = -1;

\_instance = null;

this.Dispose();

}

}

##### 密码修改ToolStripMenuItem\_Click

密码修改按钮被点击

private void 密码修改ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (!changePassword.isExist())

{

Form \_changePassword = changePassword.CreateInstance();

\_changePassword.ShowDialog();

}

}

##### 新书入库ToolStripMenuItem\_Click

新书入库按钮被点击

private void 新书入库ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (!bookAdd.isExist())

{

Form \_bookAdd = bookAdd.CreateInstance();

\_bookAdd.Show();

}

}

##### dgv\_Actions\_CellMouseUp

鼠标在datagridview的表格中弹起

private void dgv\_Actions\_CellMouseUp(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == System.Windows.Forms.MouseButtons.Right)

{

if (e.RowIndex != -1)

{

this.dgv\_Actions.Rows[e.RowIndex].Selected = true;//是否选中当前行

index = e.RowIndex;

this.dgv\_Actions.CurrentCell = this.dgv\_Actions.Rows[e.RowIndex].Cells[0];

//每次选中行都刷新到datagridview中的活动单元格

this.contextMenuStrip.Show(this.dgv\_Actions, e.Location);

//指定控件（DataGridView），指定位置（鼠标指定位置）

this.contextMenuStrip.Show(Cursor.Position);//锁定右键列表出现的位置

toolStripStatusLabel.Text = $"[Debug] 您当前操作的是第 {e.RowIndex + 1} 个UserAction，总共 {ActionList.Count} 个，{ActionList[e.RowIndex].actionDescription}";

}

}

}

##### 放弃全部操作ToolStripMenuItem\_Click

放弃全部操作按钮被点击

private void 放弃全部操作ToolStripMenuItem\_Click(object sender, EventArgs e)

{

for (int i = 0; i < ActionList.Count; i++)

{

if (ActionList[i].actionSource == "Borrow" && ActionList[i].actionType == "Borrow")

{

BorrowNotReturn--;

}

if (ActionList[i].actionSource == "Borrow" && ActionList[i].actionType == "Return")

{

BorrowNotReturn++;

}

ActionList.Remove(ActionList[i]);

i--; // ActionList.Count在上一步中减小了1，所以这里要让i不变！

}

toolStripStatusLabel.Text = $"[Info] 已放弃全部操作，就绪";

}

##### 放弃ToolStripMenuItem\_Click

右键菜单放弃按钮被点击

private void 放弃ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (index != -1)

{

if (ActionList[index].actionSource == "Borrow" && ActionList[index].actionType == "Borrow")

{

BorrowNotReturn--;

}

if (ActionList[index].actionSource == "Borrow" && ActionList[index].actionType == "Return")

{

BorrowNotReturn++;

}

toolStripStatusLabel.Text = $"[Info] 放弃操作 {ActionList[index].actionDescription} 成功，就绪";

ActionList.RemoveAt(index);

index = -1;

}

}

##### 提交操作ToolStripMenuItem1\_Click

右键菜单提交操作按钮被点击

private void 提交操作ToolStripMenuItem1\_Click(object sender, EventArgs e)

{

if (index != -1)

{

if (ActionList[index].actionModel != null)

{

if (mainControler.SubmitAction(ActionList[index]) == 0)

{

MessageBox.Show($"提交操作 {ActionList[index].actionDescription} 失败！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

toolStripStatusLabel.Text = $"[Info] 提交操作 {ActionList[index].actionDescription} 失败，就绪";

}

else

{

if (ActionList[index].actionSource == "Reader" && ActionList[index].actionType == "Password")

{

Reader tmp = (Reader)ActionList[index].actionModel;

GlobalObject.reader.rdPwd = tmp.rdPwd;

}

toolStripStatusLabel.Text = $"[Info] 提交操作 {ActionList[index].actionDescription} 成功，就绪";

ActionList.RemoveAt(index);

}

}

else

{

MessageBox.Show($"提交操作 {ActionList[index].actionDescription} 失败！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

toolStripStatusLabel.Text = $"[Info] 提交操作 {ActionList[index].actionDescription} 失败，就绪";

}

index = -1;

}

}

##### 提交全部操作ToolStripMenuItem\_Click

提交全部操作按钮被点击

private void 提交全部操作ToolStripMenuItem\_Click(object sender, EventArgs e)

{

for (int i = 0; i < ActionList.Count; i++)

{

// 遍历删除，不能使用foreach，因为容器的count会发生变化！

if (ActionList[i].actionModel != null)

{

if (mainControler.SubmitAction(ActionList[i]) == 0)

{

MessageBox.Show($"提交操作 {ActionList[i].actionDescription} 失败！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

if (ActionList[i].actionSource == "Reader" && ActionList[i].actionType == "Password")

{

Reader tmp = (Reader)ActionList[i].actionModel;

GlobalObject.reader.rdPwd = tmp.rdPwd;

}

ActionList.Remove(ActionList[i]);

i--; // ActionList.Count在上一步中减小了1，所以这里要让i不变！

}

}

else

{

MessageBox.Show($"提交操作 {ActionList[i].actionDescription} 失败！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

if (ActionList.Count != 0)

{

toolStripStatusLabel.Text = $"[Info] 已提交全部操作，但部分操作提交失败，就绪";

}

else

{

toolStripStatusLabel.Text = $"[Info] 已提交全部操作，就绪";

}

}

##### dgv\_Actions\_CellMouseDoubleClick

datagridview中表格被双击，提交被双击的用户操作

private void dgv\_Actions\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == System.Windows.Forms.MouseButtons.Left)

{

if (e.RowIndex != -1)

{

index = e.RowIndex;

if (ActionList[index].actionModel != null)

{

if (mainControler.SubmitAction(ActionList[index]) == 0)

{

MessageBox.Show($"提交操作 {ActionList[index].actionDescription} 失败！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

toolStripStatusLabel.Text = $"[Info] 提交操作 {ActionList[index].actionDescription} 失败，就绪";

}

else

{

toolStripStatusLabel.Text = $"[Info] 提交操作 {ActionList[index].actionDescription} 成功，就绪";

if (ActionList[index].actionSource == "Reader" && ActionList[index].actionType == "Password")

{

Reader tmp = (Reader)ActionList[index].actionModel;

GlobalObject.reader.rdPwd = tmp.rdPwd;

}

ActionList.RemoveAt(index);

}

}

else

{

MessageBox.Show($"提交操作 {ActionList[index].actionDescription} 失败！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

toolStripStatusLabel.Text = $"[Info] 提交操作 {ActionList[index].actionDescription} 失败，就绪";

}

index = -1;

}

}

}

##### dgv\_Actions\_CellMouseClick

datagridview中表格被单击

private void dgv\_Actions\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == System.Windows.Forms.MouseButtons.Left)

{

toolStripStatusLabel.Text = "[Info] 等待操作，就绪";

}

}

##### 图书维护ToolStripMenuItem\_Click

图书维护按钮被点击

private void 图书维护ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (!bookSearch.isExist())

{

Form \_bookSearch = bookSearch.CreateInstance();

\_bookSearch.Show();

}

}

##### 借书ToolStripMenuItem\_Click

借书按钮被点击

private void 借书ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (!bookSearch\_reader.isExist())

{

Form \_bookSearch\_available = bookSearch\_reader.CreateInstance();

\_bookSearch\_available.Show();

}

}

##### 续借ToolStripMenuItem\_Click

续借按钮被点击

private void 续借ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (!borrowSearch.isExist())

{

borrowSearch.isReturnMode = false;

Form \_borrowSearch = borrowSearch.CreateInstance();

\_borrowSearch.Show();

}

}

##### 还书ToolStripMenuItem\_Click

还书按钮被点击

private void 还书ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (!borrowSearch.isExist())

{

borrowSearch.isReturnMode = true;

Form \_borrowSearch = borrowSearch.CreateInstance();

\_borrowSearch.Show();

}

}

##### 权限管理ToolStripMenuItem\_Click

权限管理按钮被点击

private void 权限管理ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (!readerSearch.isExist())

{

readerSearch.isRBACMode = true;

Form \_readerSearch = readerSearch.CreateInstance();

\_readerSearch.Show();

}

}

#### bookAdd

对应4.8.3 添加图书窗口

##### setBookInfo

该方法用于设置图书信息，具体代码如下：

private void setBookInfo()

{

GlobalObject.bookSource.bkCode = textBox\_bkCode.Text;

GlobalObject.bookSource.bkName = textBox\_bkName.Text;

GlobalObject.bookSource.bkAuthor = textBox\_bkAuthor.Text;

GlobalObject.bookSource.bkPress = textBox\_bkPress.Text;

GlobalObject.bookSource.bkDatePress = dTP\_bkDatePress.Value;

GlobalObject.bookSource.bkISBN = textBox\_bkISBN.Text;

GlobalObject.bookSource.bkCatalog = comboBox\_bkCatalog.SelectedIndex.ToString(); // 这里进行了简化处理，实际并非如此简单。

GlobalObject.bookSource.bkLanguage = Int16.Parse(comboBox\_bkLanguage.SelectedIndex.ToString());

GlobalObject.bookSource.bkPages = Int32.Parse(textBox\_bkPages.Text);

GlobalObject.bookSource.bkPrice = decimal.Parse(textBox\_bkPrice.Text);

GlobalObject.bookSource.bkDateIn = dTP\_bkDateIn.Value;

GlobalObject.bookSource.bkBrief = textBox\_bkBrief.Text;

GlobalObject.bookSource.bkCover = bookAddControler.ImageToByte(pic\_bkCover.Image);

GlobalObject.bookSource.bkStatus = "在馆";

GlobalObject.actionSource.actionSource = "Book";

GlobalObject.actionSource.actionModel = GlobalObject.bookSource;

GlobalObject.actionSource.actionType = "Add";

}

##### btnAdd\_Click

添加按钮被点击

private void btnAdd\_Click(object sender, EventArgs e)

{

if (textBox\_bkCode.Text == "")

{

MessageBox.Show("图书编号不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_bkName.Text == "")

{

MessageBox.Show("图书名称不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_bkAuthor.Text == "")

{

MessageBox.Show("图书作者不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_bkPress.Text == "")

{

MessageBox.Show("出版社名不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_bkISBN.Text == "")

{

MessageBox.Show("ISBN不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

try

{

if (Int32.Parse(textBox\_bkPages.Text) <= 0)

{

MessageBox.Show("图书页数必须大于0！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

}

catch (Exception)

{

MessageBox.Show("请输入正确的图书页数！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

if (decimal.Parse(textBox\_bkPrice.Text) <= 0)

{

MessageBox.Show("图书价格必须大于0！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

}

catch (Exception)

{

MessageBox.Show("请输入正确的价格！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

if (Int32.Parse(textBox\_quantity.Text) <= 0)

{

MessageBox.Show("图书本书必须大于0！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

}

catch (Exception)

{

MessageBox.Show("请输入正确的图书本书！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (textBox\_bkBrief.Text == "")

{

MessageBox.Show("内容简介不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

setBookInfo();

// 如果还存在未提交的添加图书操作，则bkID从未提交操作中获取！

int index = -1;

foreach (UserAction action in main.ActionList)

{

if (action.actionSource == "Book" && action.actionType == "Add")

{

Book tmp = (Book)action.actionModel;

index = tmp.bkID;

}

}

for (int i = 0; i < Int32.Parse(textBox\_quantity.Text); i++)

{

if (index != -1)

{

textBox\_bkID.Text = (index + 1).ToString();

}

GlobalObject.bookSource.bkID = Int32.Parse(textBox\_bkID.Text) + i;

GlobalObject.actionSource.actionDescription = $"添加图书 {GlobalObject.bookSource.bkCode}(bkID:{GlobalObject.bookSource.bkID}) 。";

main.addAction(GlobalObject.actionSource);

}

this.Visible = false;

\_instance = null;

this.Dispose();

}

##### bookAdd\_Load

本窗口加载时执行

private void bookAdd\_Load(object sender, EventArgs e)

{

int bkID = bookAddControler.GetLastBkID();

if (bkID == -1)

{

MessageBox.Show("连接数据库失败，请检查您的网络连接！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

\_instance = null;

this.Dispose();

}

textBox\_bkID.Text = (bkID + 1).ToString();

comboBox\_bkCatalog.SelectedIndex = 0;

comboBox\_bkLanguage.SelectedIndex = 0;

}

##### textBox\_quantity\_KeyPress、textBox\_bkPrice\_KeyPress、textBox\_bkPages\_KeyPress

用于筛选用户输入的内容

private void textBox\_quantity\_KeyPress(object sender, KeyPressEventArgs e)

{

// 只能输入数字！ (char)8表示退格键

if (!(Char.IsNumber(e.KeyChar)) && e.KeyChar != (char)8)

{

e.Handled = true;

}

else

{

e.Handled = false;

}

}

private void textBox\_bkPrice\_KeyPress(object sender, KeyPressEventArgs e)

{

// 只能输入数字！ (char)8表示退格键 (char)46表示小数点

if (!(Char.IsNumber(e.KeyChar)) && e.KeyChar != (char)8 && e.KeyChar != (char)46)

{

e.Handled = true;

}

else

{

e.Handled = false;

}

}

private void textBox\_bkPages\_KeyPress(object sender, KeyPressEventArgs e)

{

// 只能输入数字！ (char)8表示退格键

if (!(Char.IsNumber(e.KeyChar)) && e.KeyChar != (char)8)

{

e.Handled = true;

}

else

{

e.Handled = false;

}

}

##### button\_bkCover\_Click

选择图片被点击

private void button\_bkCover\_Click(object sender, EventArgs e)

{

OpenFileDialog file = new OpenFileDialog();

file.InitialDirectory = ".";

file.Filter = "支持的图片格式 (\*.jpg, \*.jpeg, \*.png, \*.bmp)|\*.jpg;\*.jpeg;\*.png;\*.bmp|JPEG 格式|\*.jpg;\*.jpeg|PNG 格式|\*.png|BITMAP 格式|\*.bmp";

file.ShowDialog();

if (file.FileName != string.Empty)

{

try

{

this.pic\_bkCover.Load(file.FileName);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

#### bookDetail

对应4.8.4 图书详细窗口

##### bookDetail\_Load

当前窗口加载时执行

private void bookDetail\_Load(object sender, EventArgs e)

{

Book temp = bookDetailControler.GetBookbybkID(bookSearch.bkID);

if (temp == null)

{

temp = bookDetailControler.GetBookbybkID(bookSearch\_reader.bkID);

readonlymode();

}

if (temp == null)

{

MessageBox.Show("连接数据库失败，请检查您的网络连接！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

\_instance = null;

this.Dispose();

return;

}

textBox\_bkID.Text = temp.bkID.ToString();

textBox\_bkCode.Text = temp.bkCode;

textBox\_bkName.Text = temp.bkName;

textBox\_bkAuthor.Text = temp.bkAuthor;

textBox\_bkPress.Text = temp.bkPress;

dTP\_bkDatePress.Value = temp.bkDatePress;

textBox\_bkISBN.Text = temp.bkISBN;

comboBox\_bkCatalog.SelectedIndex = Int32.Parse(temp.bkCatalog);

comboBox\_bkLanguage.SelectedIndex = temp.bkLanguage;

textBox\_bkPages.Text = temp.bkPages.ToString();

textBox\_bkPrice.Text = temp.bkPrice.ToString();

dTP\_bkDateIn.Value = temp.bkDateIn;

textBox\_bkBrief.Text = temp.bkBrief;

pic\_bkCover.Image = bookDetailControler.ByteToImage(temp.bkCover);

comboBox\_Status.SelectedItem = temp.bkStatus;

}

##### Readonlymode

该方法用于切换为只读模式，如果非图书维护时需要展示图书信息，则需要执行此方法，具体代码如下：

private void readonlymode()

{

this.Text = "图书详细信息";

textBox\_bkID.Enabled = false;

textBox\_bkCode.Enabled = false;

textBox\_bkName.Enabled = false;

textBox\_bkAuthor.Enabled = false;

textBox\_bkPress.Enabled = false;

dTP\_bkDatePress.Enabled = false;

textBox\_bkISBN.Enabled = false;

comboBox\_bkCatalog.Enabled = false;

comboBox\_bkLanguage.Enabled = false;

textBox\_bkPages.Enabled = false;

textBox\_bkPrice.Enabled = false;

dTP\_bkDateIn.Enabled = false;

textBox\_bkBrief.Enabled = false;

comboBox\_Status.Enabled = false;

button\_bkCover.Enabled = false;

button\_bkCover.Visible = false;

button\_Change.Enabled = false;

button\_Change.Visible = false;

button\_Cancel.Text = "关闭";

}

##### setBookInfo

该方法用于设置图书信息，具体代码如下：

private void setBookInfo()

{

GlobalObject.bookSource.bkID = Int32.Parse(textBox\_bkID.Text);

GlobalObject.bookSource.bkCode = textBox\_bkCode.Text;

GlobalObject.bookSource.bkName = textBox\_bkName.Text;

GlobalObject.bookSource.bkAuthor = textBox\_bkAuthor.Text;

GlobalObject.bookSource.bkPress = textBox\_bkPress.Text;

GlobalObject.bookSource.bkDatePress = dTP\_bkDatePress.Value;

GlobalObject.bookSource.bkISBN = textBox\_bkISBN.Text;

GlobalObject.bookSource.bkCatalog = comboBox\_bkCatalog.SelectedIndex.ToString(); // 这里进行了简化处理，实际并非如此简单。

GlobalObject.bookSource.bkLanguage = Int16.Parse(comboBox\_bkLanguage.SelectedIndex.ToString());

GlobalObject.bookSource.bkPages = Int32.Parse(textBox\_bkPages.Text);

GlobalObject.bookSource.bkPrice = decimal.Parse(textBox\_bkPrice.Text);

GlobalObject.bookSource.bkDateIn = dTP\_bkDateIn.Value;

GlobalObject.bookSource.bkBrief = textBox\_bkBrief.Text;

GlobalObject.bookSource.bkCover = bookDetailControler.ImageToByte(pic\_bkCover.Image);

GlobalObject.bookSource.bkStatus = comboBox\_Status.SelectedItem.ToString();

GlobalObject.actionSource.actionDescription = $"修改图书信息 {GlobalObject.bookSource.bkCode}(bkID:{GlobalObject.bookSource.bkID}) ";

GlobalObject.actionSource.actionSource = "Book";

GlobalObject.actionSource.actionModel = GlobalObject.bookSource;

GlobalObject.actionSource.actionType = "Update";

}

##### btnChange\_Click

修改按钮被点击

private void btnChange\_Click(object sender, EventArgs e)

{

if (textBox\_bkCode.Text == "")

{

MessageBox.Show("图书编号不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_bkName.Text == "")

{

MessageBox.Show("图书名称不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_bkAuthor.Text == "")

{

MessageBox.Show("图书作者不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_bkPress.Text == "")

{

MessageBox.Show("出版社名不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_bkISBN.Text == "")

{

MessageBox.Show("ISBN不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

try

{

if (Int32.Parse(textBox\_bkPages.Text) <= 0)

{

MessageBox.Show("图书页数必须大于0！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

}

catch (Exception)

{

MessageBox.Show("请输入正确的图书页数！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

if (decimal.Parse(textBox\_bkPrice.Text) <= 0)

{

MessageBox.Show("图书价格必须大于0！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

}

catch (Exception)

{

MessageBox.Show("请输入正确的价格！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (textBox\_bkBrief.Text == "")

{

MessageBox.Show("内容简介不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

setBookInfo();

// 如果该图书还存在未提交的修改图书操作，报错！

foreach (UserAction action in main.ActionList)

{

if (action.actionSource == "Book" && action.actionType == "Update")

{

Book tmp = (Book)action.actionModel;

if (tmp.bkID == GlobalObject.bookSource.bkID)

{

MessageBox.Show("对当前图书的修改操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

}

main.addAction(GlobalObject.actionSource);

this.Visible = false;

this.DialogResult = DialogResult.OK;

\_instance = null;

this.Dispose();

}

#### bookSearch

对应4.8.5 图书选择窗口

##### 属性

private static readonly int \_maxrow = 20;

public static int bkID = -1;

private int index = -1;

private int target = -1;

##### dvg\_style

该方法用于设置datagridview的样式，具体代码如下：

private void dvg\_style()

{

if (dgv\_target.RowCount != 0)

{

int h = (dgv\_target.Size.Height - dgv\_target.ColumnHeadersHeight) / \_maxrow;

for (int i = 0; i < dgv\_target.RowCount; i++)

{

dgv\_target.Rows[i].Height = h;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == "0")

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Blue;

}

foreach (UserAction action in main.ActionList)

{

if (action.actionSource == "Book" && action.actionType == "Delete")

{

Book tmp = (Book)action.actionModel;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == tmp.bkID.ToString())

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Red;

}

}

if (action.actionSource == "Book" && action.actionType == "Update")

{

Book tmp = (Book)action.actionModel;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == tmp.bkID.ToString())

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Orange;

}

}

}

}

}

dgv\_target.Columns["图书序号"].FillWeight = 8;

dgv\_target.Columns["图书编号"].FillWeight = 8;

dgv\_target.Columns["图书名称"].FillWeight = 20;

dgv\_target.Columns["图书作者"].FillWeight = 10;

dgv\_target.Columns["出版社名"].FillWeight = 15;

dgv\_target.Columns["标准ISBN"].FillWeight = 15;

dgv\_target.Columns["出版日期"].FillWeight = 15;

dgv\_target.Columns["图书状态"].FillWeight = 9;

}

##### button\_check

该方法用于判断上一页、下一页按钮是否可以点击，具体代码如下：

private void button\_check()

{

if (index == 1)

{

button\_previous.Enabled = false;

}

else

{

button\_previous.Enabled = true;

}

if (dgv\_target.RowCount < \_maxrow)

{

button\_next.Enabled = false;

}

else

{

button\_next.Enabled = true;

}

}

##### bookSearch\_Load

窗口加载时执行

private void bookSearch\_Load(object sender, EventArgs e)

{

toolStripComboBox\_type.SelectedIndex = 0;

dgv\_target.DataSource = bookSearchControler.GetDTby("\_ALL", "", 0, \_maxrow);

index = 1;

dvg\_style();

button\_check();

label\_info.Text = "[Info] 等待选择，就绪";

}

##### toolStripButton\_search\_Click

查找按钮被点击

private void toolStripButton\_search\_Click(object sender, EventArgs e)

{

string type;

switch (toolStripComboBox\_type.SelectedIndex)

{

case 0:

type = "bkID";

break;

case 1:

type = "bkCode";

break;

case 2:

type = "bkName";

break;

case 3:

type = "bkAuthor";

break;

case 4:

type = "bkPress";

break;

case 5:

type = "bkISBN";

break;

default:

type = "\_ALL";

break;

}

if (toolStripTextBox\_text.Text == "")

{

type = "\_ALL";

}

try

{

dgv\_target.DataSource = bookSearchControler.GetDTby(type, toolStripTextBox\_text.Text, 0, \_maxrow);

}

catch (Exception)

{

MessageBox.Show("请输入正确查询内容！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

index = 1;

dvg\_style();

button\_check();

label\_info.Text = "[Info] 查找完成，就绪";

}

##### button\_next\_Click

下一页按钮被点击

private void button\_next\_Click(object sender, EventArgs e)

{

string type;

switch (toolStripComboBox\_type.SelectedIndex)

{

case 0:

type = "bkID";

break;

case 1:

type = "bkCode";

break;

case 2:

type = "bkName";

break;

case 3:

type = "bkAuthor";

break;

case 4:

type = "bkPress";

break;

case 5:

type = "bkISBN";

break;

default:

type = "\_ALL";

break;

}

if (toolStripTextBox\_text.Text == "")

{

type = "\_ALL";

}

try

{

dgv\_target.DataSource = bookSearchControler.GetDTby(type, toolStripTextBox\_text.Text, ((++index) - 1) \* \_maxrow, \_maxrow);

}

catch (Exception)

{

MessageBox.Show("请输入正确查询内容！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

dvg\_style();

button\_check();

}

##### button\_previous\_Click

上一页按钮被点击

private void button\_previous\_Click(object sender, EventArgs e)

{

string type;

switch (toolStripComboBox\_type.SelectedIndex)

{

case 0:

type = "bkID";

break;

case 1:

type = "bkCode";

break;

case 2:

type = "bkName";

break;

case 3:

type = "bkAuthor";

break;

case 4:

type = "bkPress";

break;

case 5:

type = "bkISBN";

break;

default:

type = "\_ALL";

break;

}

if (toolStripTextBox\_text.Text == "")

{

type = "\_ALL";

}

try

{

dgv\_target.DataSource = bookSearchControler.GetDTby(type, toolStripTextBox\_text.Text, ((--index) - 1) \* \_maxrow, \_maxrow);

}

catch (Exception)

{

MessageBox.Show("请输入正确查询内容！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

dvg\_style();

button\_check();

}

##### bookSearch\_Resize

窗口大小改变时

private void bookSearch\_Resize(object sender, EventArgs e)

{

if (dgv\_target.RowCount != 0)

{

int h = (dgv\_target.Size.Height - dgv\_target.ColumnHeadersHeight) / \_maxrow;

for (int i = 0; i < dgv\_target.RowCount; i++)

{

dgv\_target.Rows[i].Height = h;

}

}

}

##### dgv\_target\_CellMouseUp

鼠标在datagridview的表格中弹起

private void dgv\_target\_CellMouseUp(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == System.Windows.Forms.MouseButtons.Right)

{

if (e.RowIndex != -1)

{

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Blue)

{

MessageBox.Show("该图书属于系统，禁止对其进行任何操作！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Red)

{

MessageBox.Show("对当前图书的删除操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Orange)

{

MessageBox.Show("对当前图书的修改操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

this.dgv\_target.Rows[e.RowIndex].Selected = true;//是否选中当前行

target = e.RowIndex;

this.dgv\_target.CurrentCell = this.dgv\_target.Rows[e.RowIndex].Cells[0];

//每次选中行都刷新到datagridview中的活动单元格

this.contextMenuStrip.Show(this.dgv\_target, e.Location);

//指定控件（DataGridView），指定位置（鼠标指定位置）

this.contextMenuStrip.Show(Cursor.Position);//锁定右键列表出现的位置

label\_info.Text = $"[Debug] 您当前操作的图书是 bkCode:{dgv\_target.Rows[e.RowIndex].Cells[1].Value}(bkID:{dgv\_target.Rows[e.RowIndex].Cells[0].Value})";

}

}

}

##### 删除图书ToolStripMenuItem\_Click

删除图书按钮被点击

private void 删除图书ToolStripMenuItem\_Click(object sender, EventArgs e)

{

GlobalObject.bookSource.bkID = Int32.Parse(dgv\_target.Rows[target].Cells[0].Value.ToString());

GlobalObject.bookSource.bkCode = dgv\_target.Rows[target].Cells[1].Value.ToString();

GlobalObject.actionSource.actionSource = "Book";

GlobalObject.actionSource.actionModel = GlobalObject.bookSource;

GlobalObject.actionSource.actionType = "Delete";

GlobalObject.actionSource.actionDescription = $"删除图书 {GlobalObject.bookSource.bkCode}(bkID:{GlobalObject.bookSource.bkID}) 。";

main.addAction(GlobalObject.actionSource);

dgv\_target.Rows[target].DefaultCellStyle.BackColor = Color.Red;

label\_info.Text = $"[Info] 删除图书 bkCode:{dgv\_target.Rows[target].Cells[1].Value}(bkID:{dgv\_target.Rows[target].Cells[0].Value}) 已挂起，就绪";

}

##### 编辑图书信息ToolStripMenuItem\_Click

编辑图书信息按钮被点击

private void 编辑图书信息ToolStripMenuItem\_Click(object sender, EventArgs e)

{

bkID = Int32.Parse(dgv\_target.Rows[target].Cells[0].Value.ToString());

Form \_bookDetail = bookDetail.CreateInstance();

if (\_bookDetail.ShowDialog() == DialogResult.OK)

{

bkID = -1;

dgv\_target.Rows[target].DefaultCellStyle.BackColor = Color.Orange;

label\_info.Text = $"[Info] 修改图书信息 bkCode:{dgv\_target.Rows[target].Cells[1].Value}(bkID:{dgv\_target.Rows[target].Cells[0].Value}) 已挂起，就绪";

}

}

##### dgv\_target\_CellMouseDoubleClick

datagridview中表格被双击，编辑被双击的图书信息

private void dgv\_target\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == System.Windows.Forms.MouseButtons.Left)

{

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Blue)

{

MessageBox.Show("该图书属于系统，禁止对其进行任何操作！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Red)

{

MessageBox.Show("对当前图书的删除操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Orange)

{

MessageBox.Show("对当前图书的修改操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (e.RowIndex != -1)

{

target = e.RowIndex;

bkID = Int32.Parse(dgv\_target.Rows[target].Cells[0].Value.ToString());

Form \_bookDetail = bookDetail.CreateInstance();

if (\_bookDetail.ShowDialog() == DialogResult.OK)

{

dgv\_target.Rows[target].DefaultCellStyle.BackColor = Color.Orange;

label\_info.Text = $"[Info] 修改图书信息 bkCode:{dgv\_target.Rows[target].Cells[1].Value}(bkID:{dgv\_target.Rows[target].Cells[0].Value}) 已挂起，就绪";

}

}

}

}

#### bookSearch\_reader

对应4.8.6 借书的图书选择窗口

该窗口的方法与bookSearch相似！

##### dvg\_style

该方法用于设置datagridview的样式，具体代码如下：

private void dvg\_style()

{

if (dgv\_target.RowCount != 0)

{

int h = (dgv\_target.Size.Height - dgv\_target.ColumnHeadersHeight) / \_maxrow;

for (int i = 0; i < dgv\_target.RowCount; i++)

{

dgv\_target.Rows[i].Height = h;

if (dgv\_target.Rows[i].Cells[7].Value.ToString() != "在馆")

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.LightGoldenrodYellow;

}

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == "0")

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Blue;

}

foreach (UserAction action in main.ActionList)

{

if (action.actionSource == "Book" && action.actionType == "Delete")

{

Book tmp = (Book)action.actionModel;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == tmp.bkID.ToString())

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Red;

}

}

if (action.actionSource == "Book" && action.actionType == "Update")

{

Book tmp = (Book)action.actionModel;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == tmp.bkID.ToString())

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Orange;

}

}

if (action.actionSource == "Borrow" && action.actionType == "Borrow")

{

Borrow tmp = (Borrow)action.actionModel;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == tmp.bkID.ToString())

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Green;

}

}

}

}

}

dgv\_target.Columns["图书序号"].FillWeight = 8;

dgv\_target.Columns["图书编号"].FillWeight = 8;

dgv\_target.Columns["图书名称"].FillWeight = 20;

dgv\_target.Columns["图书作者"].FillWeight = 10;

dgv\_target.Columns["出版社名"].FillWeight = 15;

dgv\_target.Columns["标准ISBN"].FillWeight = 15;

dgv\_target.Columns["出版日期"].FillWeight = 15;

dgv\_target.Columns["图书状态"].FillWeight = 9;

}

##### bookSearch\_available\_Load

当前窗口加载时执行

private void bookSearch\_available\_Load(object sender, EventArgs e)

{

toolStripComboBox\_type.SelectedIndex = 0;

dgv\_target.DataSource = bookSearchControler.GetDTby("\_ALL", "", 0, \_maxrow);

index = 1;

dvg\_style();

button\_check();

label\_info.Text = "[Info] 等待选择，就绪";

}

##### 借阅图书ToolStripMenuItem\_Click

借阅图书按钮被点击

private void 借阅图书ToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor == Color.Green)

{

MessageBox.Show("对当前图书的借阅操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor == Color.LightGoldenrodYellow)

{

MessageBox.Show("当前图书不在馆，无法借阅！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (main.BorrowNotReturn >= main.BorrowMax)

{

MessageBox.Show($"当前已借阅{main.BorrowNotReturn}本书，您最多借阅{main.BorrowMax}本书。\n请归还部分图书后再尝试借阅！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

GlobalObject.borrowSource.bkID = Int32.Parse(dgv\_target.Rows[target].Cells[0].Value.ToString());

GlobalObject.borrowSource.rdID = GlobalObject.reader.rdID;

GlobalObject.borrowSource.OperatorLend = "For Debug Only!";

GlobalObject.actionSource.actionSource = "Borrow";

GlobalObject.actionSource.actionModel = GlobalObject.borrowSource;

GlobalObject.actionSource.actionType = "Borrow";

GlobalObject.actionSource.actionDescription = $"借阅图书 {GlobalObject.borrowSource.bkID}(rdID:{GlobalObject.borrowSource.rdID})。";

main.addAction(GlobalObject.actionSource);

dgv\_target.Rows[target].DefaultCellStyle.BackColor = Color.Green;

main.BorrowNotReturn++;

label\_info.Text = $"[Info] 借阅图书 bkCode:{dgv\_target.Rows[target].Cells[1].Value}(bkID:{dgv\_target.Rows[target].Cells[0].Value}) 已挂起，就绪";

}

##### 查看图书信息ToolStripMenuItem\_Click

查看图书信息按钮被点击

private void 查看图书信息ToolStripMenuItem\_Click(object sender, EventArgs e)

{

bkID = Int32.Parse(dgv\_target.Rows[target].Cells[0].Value.ToString());

Form \_bookDetail = bookDetail.CreateInstance();

\_bookDetail.ShowDialog();

}

##### dgv\_target\_CellMouseDoubleClick

datagridview中表格被双击，借阅被双击图书

private void dgv\_target\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == System.Windows.Forms.MouseButtons.Left)

{

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Blue)

{

MessageBox.Show("该图书属于系统，禁止对其进行任何操作！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Red)

{

MessageBox.Show("对当前图书的删除操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[e.RowIndex].DefaultCellStyle.BackColor == Color.Orange)

{

MessageBox.Show("对当前图书的修改操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (e.RowIndex != -1)

{

target = e.RowIndex;

bkID = Int32.Parse(dgv\_target.Rows[target].Cells[0].Value.ToString());

Form \_bookDetail = bookDetail.CreateInstance();

\_bookDetail.ShowDialog();

}

}

}

#### borrowSearch

对应4.8.7 续借和还书窗口

该窗口的方法与bookSearch相似！

##### 属性

public static bool isReturnMode = false;

delegate void ContinueOrReturn();

private static readonly int \_maxrow = 10;

public static Int64 borrowID = -1;

private int index = -1;

private int target = -1;

##### ContinueBook

该方法为续借图书方法，具体代码如下：

private void ContinueBook()

{

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor == Color.Green)

{

MessageBox.Show("对当前图书的续借操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor == Color.Indigo)

{

MessageBox.Show("对当前图书的还书操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor != Color.LightGoldenrodYellow)

{

MessageBox.Show("您已还书，无法进行当前操作！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

GlobalObject.borrowSource.BorrowID = Int64.Parse(dgv\_target.Rows[target].Cells[0].Value.ToString());

GlobalObject.borrowSource.bkID = Int32.Parse(dgv\_target.Rows[target].Cells[1].Value.ToString());

GlobalObject.actionSource.actionSource = "Borrow";

GlobalObject.actionSource.actionModel = GlobalObject.borrowSource;

GlobalObject.actionSource.actionType = "Continue";

GlobalObject.actionSource.actionDescription = $"续借图书 {GlobalObject.borrowSource.bkID}(BorrowID:{GlobalObject.borrowSource.BorrowID} rdID:{GlobalObject.reader.rdID})。";

main.addAction(GlobalObject.actionSource);

dgv\_target.Rows[target].DefaultCellStyle.BackColor = Color.Green;

label\_info.Text = $"[Info] 续借图书 bkID:{dgv\_target.Rows[target].Cells[1].Value} 已挂起，就绪";

}

##### ReturnBook

该方法为还书方法，具体代码如下：

private void ReturnBook()

{

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor == Color.Green)

{

MessageBox.Show("对当前图书的续借操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor == Color.Indigo)

{

MessageBox.Show("对当前图书的还书操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor != Color.LightGoldenrodYellow)

{

MessageBox.Show("您已还书，无法进行当前操作！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

GlobalObject.borrowSource.BorrowID = Int64.Parse(dgv\_target.Rows[target].Cells[0].Value.ToString());

GlobalObject.borrowSource.bkID = Int32.Parse(dgv\_target.Rows[target].Cells[1].Value.ToString());

GlobalObject.actionSource.actionSource = "Borrow";

GlobalObject.actionSource.actionModel = GlobalObject.borrowSource;

GlobalObject.actionSource.actionType = "Return";

GlobalObject.actionSource.actionDescription = $"归还图书 {GlobalObject.borrowSource.bkID}(BorrowID:{GlobalObject.borrowSource.BorrowID} rdID:{GlobalObject.reader.rdID})。";

main.addAction(GlobalObject.actionSource);

main.BorrowNotReturn--;

dgv\_target.Rows[target].DefaultCellStyle.BackColor = Color.Indigo;

label\_info.Text = $"[Info] 归还图书 bkID:{dgv\_target.Rows[target].Cells[1].Value} 已挂起，就绪";

}

##### ReturnMode

该方法在还书模式时执行

private void ReturnMode()

{

this.Text = "请选择您要归还的借阅记录";

借阅此图书ToolStripMenuItem.Text = "归还此图书";

}

##### dvg\_style

该方法用于设置datagridview的样式，具体代码如下：

private void dvg\_style()

{

if (dgv\_target.RowCount != 0)

{

int h = (dgv\_target.Size.Height - dgv\_target.ColumnHeadersHeight) / \_maxrow;

for (int i = 0; i < dgv\_target.RowCount; i++)

{

dgv\_target.Rows[i].Height = h;

if (dgv\_target.Rows[i].Cells[5].Value.ToString() != "1")

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.LightGoldenrodYellow;

}

foreach (UserAction action in main.ActionList)

{

if (action.actionSource == "Borrow" && action.actionType == "Continue")

{

Borrow tmp = (Borrow)action.actionModel;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == tmp.BorrowID.ToString())

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Green;

}

}

if (action.actionSource == "Borrow" && action.actionType == "Return")

{

Borrow tmp = (Borrow)action.actionModel;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == tmp.BorrowID.ToString())

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Indigo;

}

}

}

}

}

dgv\_target.Columns["借书顺序号"].FillWeight = 14;

dgv\_target.Columns["图书序号"].FillWeight = 12;

dgv\_target.Columns["续借次数"].FillWeight = 12;

dgv\_target.Columns["借书日期"].FillWeight = 15;

dgv\_target.Columns["应还日期"].FillWeight = 15;

dgv\_target.Columns["是否还书"].FillWeight = 12;

dgv\_target.Columns["借书操作员"].FillWeight = 20;

}

##### borrowSearch\_Load

窗口加载时执行

private void borrowSearch\_Load(object sender, EventArgs e)

{

if (isReturnMode)

{

ReturnMode();

}

toolStripComboBox\_type.SelectedIndex = 0;

dgv\_target.DataSource = borrowSearchControler.GetDTby("\_ALL", "", 0, \_maxrow, GlobalObject.reader.rdID);

index = 1;

dvg\_style();

button\_check();

label\_info.Text = "[Info] 等待选择，就绪";

}

##### toolStripButton\_search\_Click

查找按钮被点击

private void toolStripButton\_search\_Click(object sender, EventArgs e)

{

string type;

switch (toolStripComboBox\_type.SelectedIndex)

{

case 0:

type = "BorrowID";

break;

case 1:

type = "bkID";

break;

case 2:

type = "OperatorLend";

break;

default:

type = "\_ALL";

break;

}

if (toolStripTextBox\_text.Text == "")

{

type = "\_ALL";

}

try

{

dgv\_target.DataSource = borrowSearchControler.GetDTby(type, toolStripTextBox\_text.Text, 0, \_maxrow, GlobalObject.reader.rdID);

}

catch (Exception)

{

MessageBox.Show("请输入正确查询内容！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

index = 1;

dvg\_style();

button\_check();

label\_info.Text = "[Info] 查找完成，就绪";

}

##### button\_next\_Click

下一页按钮被点击

private void button\_next\_Click(object sender, EventArgs e)

{

string type;

switch (toolStripComboBox\_type.SelectedIndex)

{

case 0:

type = "BorrowID";

break;

case 1:

type = "bkID";

break;

case 2:

type = "OperatorLend";

break;

default:

type = "\_ALL";

break;

}

if (toolStripTextBox\_text.Text == "")

{

type = "\_ALL";

}

try

{

dgv\_target.DataSource = borrowSearchControler.GetDTby(type, toolStripTextBox\_text.Text, ((++index) - 1) \* \_maxrow, \_maxrow, GlobalObject.reader.rdID);

}

catch (Exception)

{

MessageBox.Show("请输入正确查询内容！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

dvg\_style();

button\_check();

}

##### button\_previous\_Click

上一页按钮被点击

private void button\_previous\_Click(object sender, EventArgs e)

{

string type;

switch (toolStripComboBox\_type.SelectedIndex)

{

case 0:

type = "BorrowID";

break;

case 1:

type = "bkID";

break;

case 2:

type = "OperatorLend";

break;

default:

type = "\_ALL";

break;

}

if (toolStripTextBox\_text.Text == "")

{

type = "\_ALL";

}

try

{

dgv\_target.DataSource = borrowSearchControler.GetDTby(type, toolStripTextBox\_text.Text, ((--index) - 1) \* \_maxrow, \_maxrow, GlobalObject.reader.rdID);

}

catch (Exception)

{

MessageBox.Show("请输入正确查询内容！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

dvg\_style();

button\_check();

}

##### 续借图书ToolStripMenuItem\_Click

续借（还书）按钮被点击

private void 续借图书ToolStripMenuItem\_Click(object sender, EventArgs e)

{

ContinueOrReturn x;

if (isReturnMode)

{

x = new ContinueOrReturn(ReturnBook);

}

else

{

x = new ContinueOrReturn(ContinueBook);

}

x();

}

#### changePassword

对应4.8.8 修改密码窗口

##### button\_OK\_Click

确认按钮被点击

private void button\_OK\_Click(object sender, EventArgs e)

{

foreach (UserAction action in main.ActionList)

{

if (action.actionSource == "Reader" && action.actionType == "Password")

{

Reader tmp = (Reader)action.actionModel;

if (tmp.rdID == GlobalObject.reader.rdID)

{

MessageBox.Show("当前操作列表中已存在对当前用户的修改，请检查！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

}

if (textBox\_old\_pwd.Text == "")

{

MessageBox.Show("原密码不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_new\_pwd.Text == "")

{

MessageBox.Show("新密码不能为空！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_new\_pwd.Text != textBox\_new\_pwd2.Text)

{

MessageBox.Show("两次输入的密码不一致！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (GlobalObject.reader.rdPwd != EncryptProvider.Md5(textBox\_old\_pwd.Text))

{

MessageBox.Show("原密码错误！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

if (textBox\_new\_pwd.Text == textBox\_old\_pwd.Text)

{

MessageBox.Show("新密码不能与旧密码一样！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

GlobalObject.readerSource.rdID = GlobalObject.reader.rdID;

GlobalObject.readerSource.rdName = GlobalObject.reader.rdName;

GlobalObject.readerSource.rdPwd = EncryptProvider.Md5(textBox\_new\_pwd.Text);

GlobalObject.actionSource.actionSource = "Reader";

GlobalObject.actionSource.actionModel = GlobalObject.readerSource;

GlobalObject.actionSource.actionType = "Password";

GlobalObject.actionSource.actionDescription = $"修改用户 {GlobalObject.readerSource.rdName}(rdID:{GlobalObject.readerSource.rdID}) 的密码";

main.addAction(GlobalObject.actionSource);

this.Visible = false;

\_instance = null;

this.Dispose();

}

#### readerSearch

对应4.8.10 管理读者窗口

该窗口的方法与bookSearch相似！

##### 属性

public static bool isRBACMode = false;

delegate void RBACorReaderManagement();

private static readonly int \_maxrow = 15;

public static Int32 rdID = -1;

private int index = -1;

private int target = -1;

##### RBACMode

该方法在权限管理模式时执行

private void RBACMode()

{

this.Text = "请选择您要进行权限管理的读者";

删除此读者ToolStripMenuItem.Text = "权限管理";

}

##### RBAC

该方法用于修改用户权限

private void RBAC()

{

if (dgv\_target.Rows[target].DefaultCellStyle.BackColor == Color.Green)

{

MessageBox.Show("对当前读者的权限修改操作已存在于操作队列！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

Form \_changePermission = changePermission.CreateInstance();

if (\_changePermission.ShowDialog() == DialogResult.OK)

{

dgv\_target.Rows[target].DefaultCellStyle.BackColor = Color.Green;

label\_info.Text = $"[Info] 修改权限 rdID:{GlobalObject.readerSource.rdID} 已挂起，就绪";

}

}

##### dvg\_style

该方法用于设置datagridview的样式，具体代码如下：

private void dvg\_style()

{

if (dgv\_target.RowCount != 0)

{

int h = (dgv\_target.Size.Height - dgv\_target.ColumnHeadersHeight) / \_maxrow;

for (int i = 0; i < dgv\_target.RowCount; i++)

{

dgv\_target.Rows[i].Height = h;

if (dgv\_target.Rows[i].Cells[5].Value.ToString() != "0")

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.LightYellow;

}

if (dgv\_target.Rows[i].Cells[4].Value.ToString() != "有效")

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.LightSkyBlue;

}

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == "0")

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Blue;

}

foreach (UserAction action in main.ActionList)

{

if (action.actionSource == "Reader" && action.actionType == "Permission")

{

Reader tmp = (Reader)action.actionModel;

if (dgv\_target.Rows[i].Cells[0].Value.ToString() == tmp.rdID.ToString())

{

dgv\_target.Rows[i].DefaultCellStyle.BackColor = Color.Green;

}

}

}

}

}

dgv\_target.Columns["读者ID"].FillWeight = 20;

dgv\_target.Columns["读者姓名"].FillWeight = 15;

dgv\_target.Columns["性别"].FillWeight = 15;

dgv\_target.Columns["电话号码"].FillWeight = 20;

dgv\_target.Columns["证件状态"].FillWeight = 15;

dgv\_target.Columns["职权"].FillWeight = 15;

}

##### readerSearch\_Load

窗口载入时执行

private void readerSearch\_Load(object sender, EventArgs e)

{

if (isRBACMode)

{

RBACMode();

}

toolStripComboBox\_type.SelectedIndex = 0;

dgv\_target.DataSource = readerSearchControler.GetDTby("\_ALL", "", 0, \_maxrow);

index = 1;

dvg\_style();

button\_check();

label\_info.Text = "[Info] 等待选择，就绪";

}

##### toolStripButton\_search\_Click

查找按钮被点击

private void toolStripButton\_search\_Click(object sender, EventArgs e)

{

string type;

switch (toolStripComboBox\_type.SelectedIndex)

{

case 0:

type = "rdID";

break;

case 1:

type = "rdName";

break;

case 2:

type = "rdSex";

break;

case 3:

type = "rdDept";

break;

case 4:

type = "rdPhone";

break;

case 5:

type = "rdEmail";

break;

case 6:

type = "rdStatus";

break;

default:

type = "\_ALL";

break;

}

if (toolStripTextBox\_text.Text == "")

{

type = "\_ALL";

}

try

{

dgv\_target.DataSource = readerSearchControler.GetDTby(type, toolStripTextBox\_text.Text, 0, \_maxrow);

}

catch (Exception)

{

MessageBox.Show("请输入正确查询内容！", "错误：", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

index = 1;

dvg\_style();

button\_check();

label\_info.Text = "[Info] 查找完成，就绪";

}

#### changePermission

对应4.8.9 修改权限窗口

##### GetPermission

该方法用于获取勾选的权限，具体代码如下：

private Int16 GetPermission()

{

string permission = "0";

if (checkBox\_borrowCardAdmin.Checked)

{

permission = permission + "1";

}

else

{

permission = permission + "0";

}

if (checkBox\_bookAdmin.Checked)

{

permission = permission + "1";

}

else

{

permission = permission + "0";

}

if (checkBox\_borrowAdmin.Checked)

{

permission = permission + "1";

}

else

{

permission = permission + "0";

}

if (checkBox\_systemAdmin.Checked)

{

permission = permission + "1";

}

else

{

permission = permission + "0";

}

return Convert.ToInt16(permission, 2);

}

##### button\_OK\_Click

修改按钮被点击

private void button\_OK\_Click(object sender, EventArgs e)

{

if (permission == GetPermission())

{

MessageBox.Show("该用户的权限没有发生变化！", "提示：", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

return;

}

GlobalObject.readerSource.rdID = readerSearch.rdID;

GlobalObject.readerSource.rdAdminRoles = GetPermission();

GlobalObject.actionSource.actionSource = "Reader";

GlobalObject.actionSource.actionModel = GlobalObject.readerSource;

GlobalObject.actionSource.actionType = "Permission";

GlobalObject.actionSource.actionDescription = $"修改读者 {GlobalObject.readerSource.rdID} 的权限。";

main.addAction(GlobalObject.actionSource);

this.Visible = false;

this.DialogResult = DialogResult.OK;

\_instance = null;

this.Dispose();

}

##### changePermission\_Load

窗口载入时执行

private void changePermission\_Load(object sender, EventArgs e)

{

Reader target = loginControler.GetReader(readerSearch.rdID);

if (target == null)

{

return;

}

permission = target.rdAdminRoles;

if ((permission & 0b1000) == 0b1000)

{

checkBox\_borrowCardAdmin.Checked = true;

}

if ((permission & 0b0100) == 0b0100)

{

checkBox\_bookAdmin.Checked = true;

}

if ((permission & 0b0010) == 0b0010)

{

checkBox\_borrowAdmin.Checked = true;

}

if ((permission & 0b0001) == 0b0001)

{

checkBox\_systemAdmin.Checked = true;

}

}

## 六、系统演示

### 6.1 登录系统

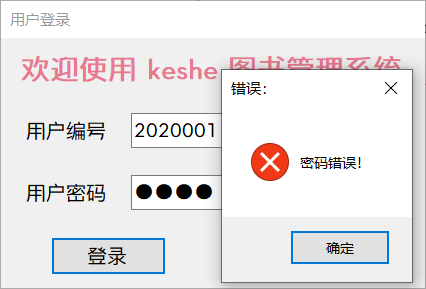


图6.1.1 密码错误

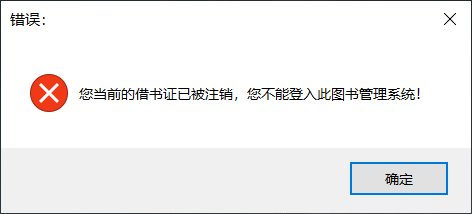


图6.1.2 借书证被注销，无法登录

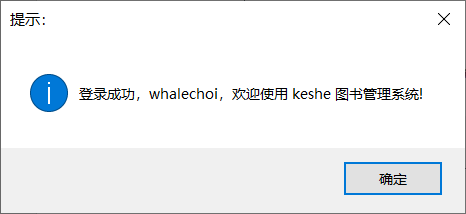


图6.1.3 登录成功

### 6.2 添加35本《c程序设计》



图6.2.1 添加图书

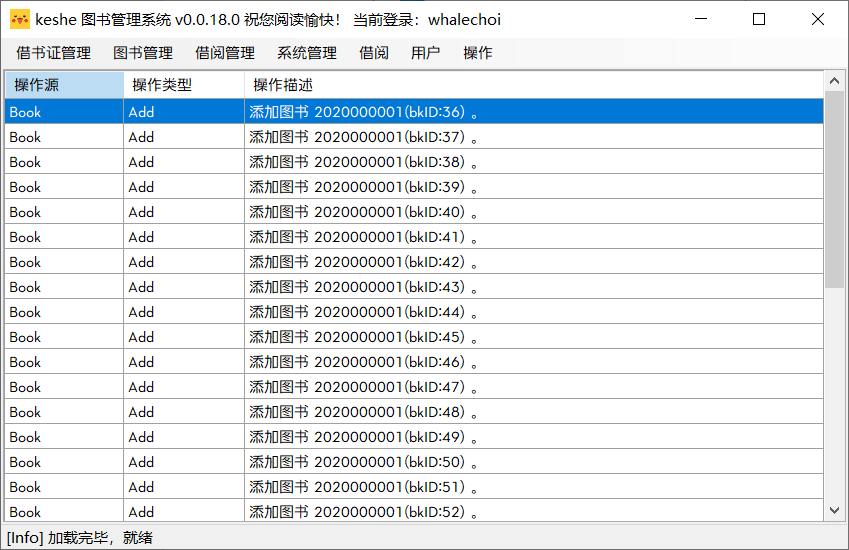


图6.2.2 操作被挂起

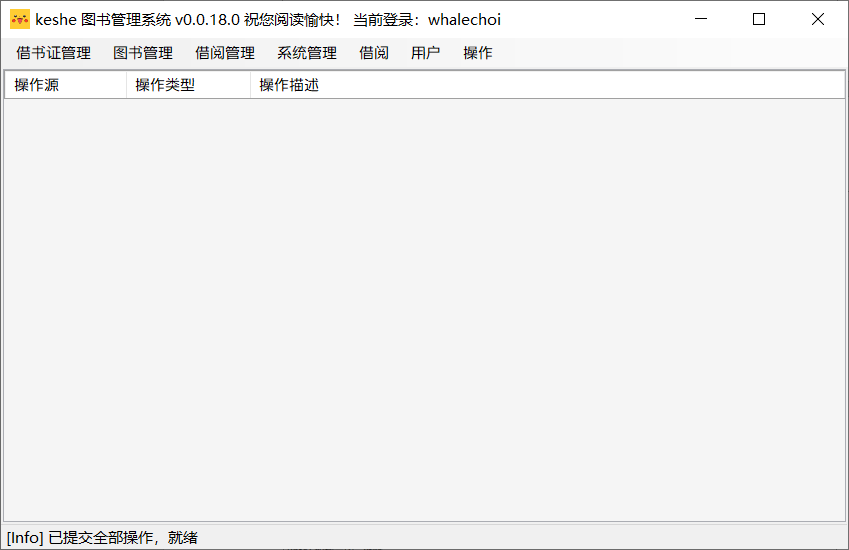


图6.2.3 提交全部操作成功

### 6.3 维护52号图书

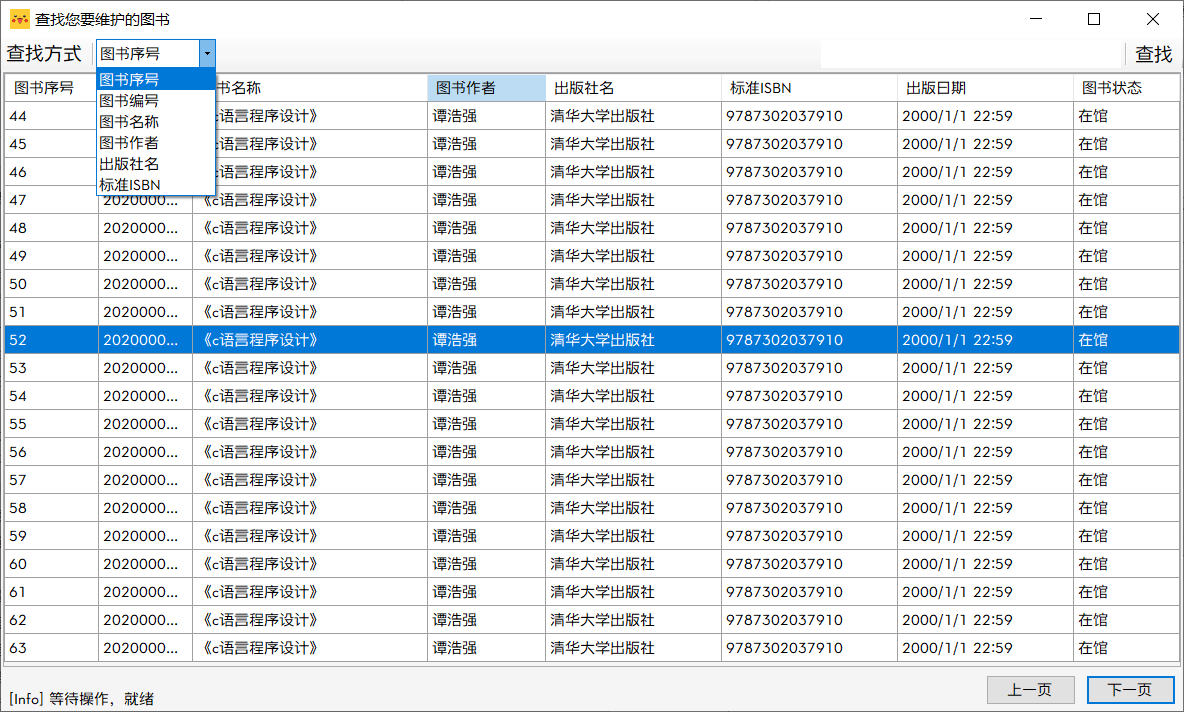


图6.3.1 找到要维护的图书



图6.3.2维护的图书

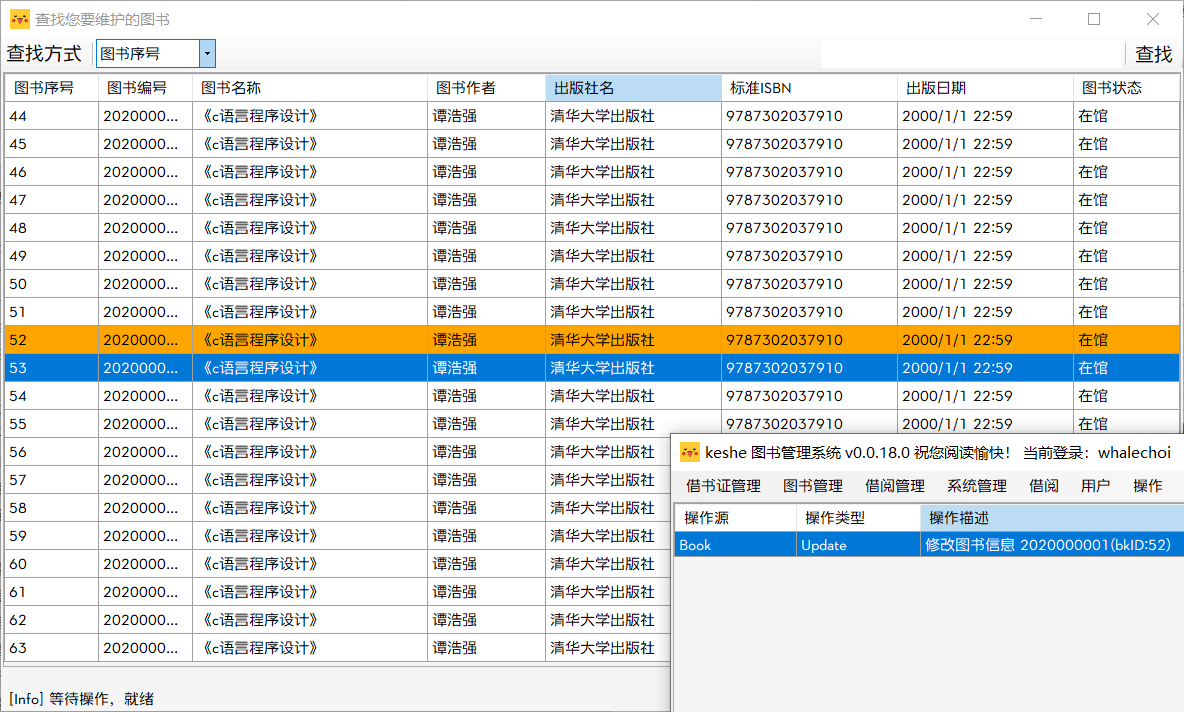


图6.3.3维护操作被挂起

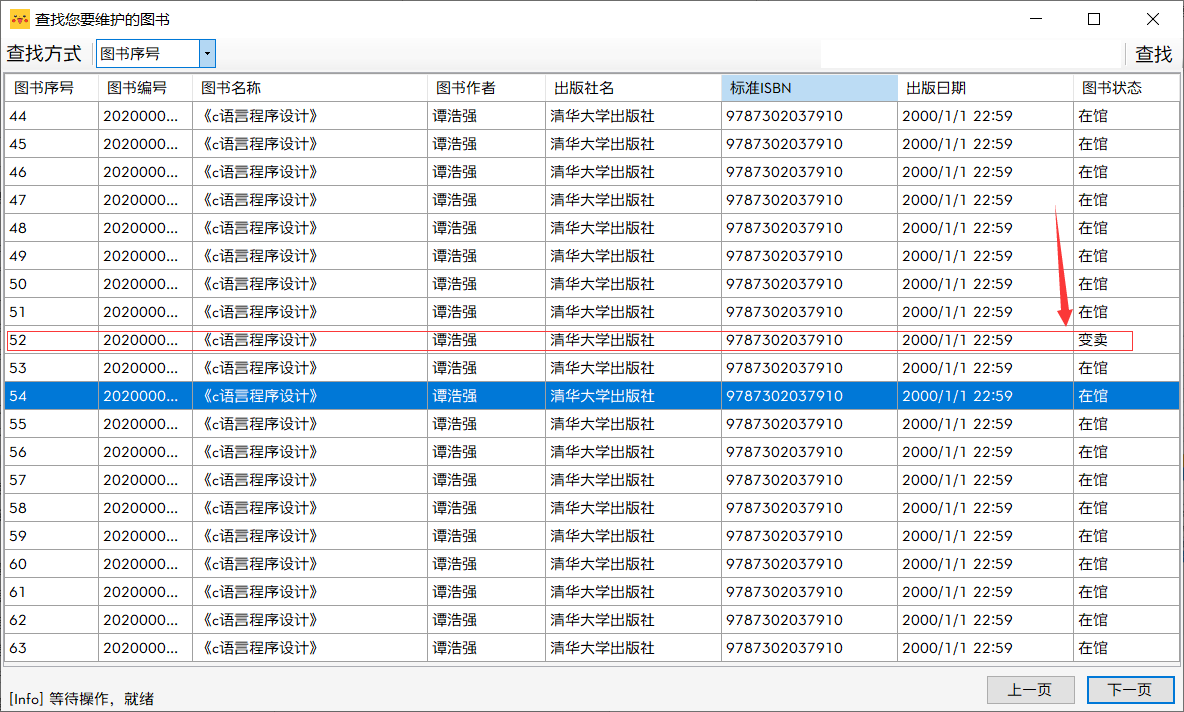


图6.3.4提交操作后维护成功

### 6.4 借阅图书

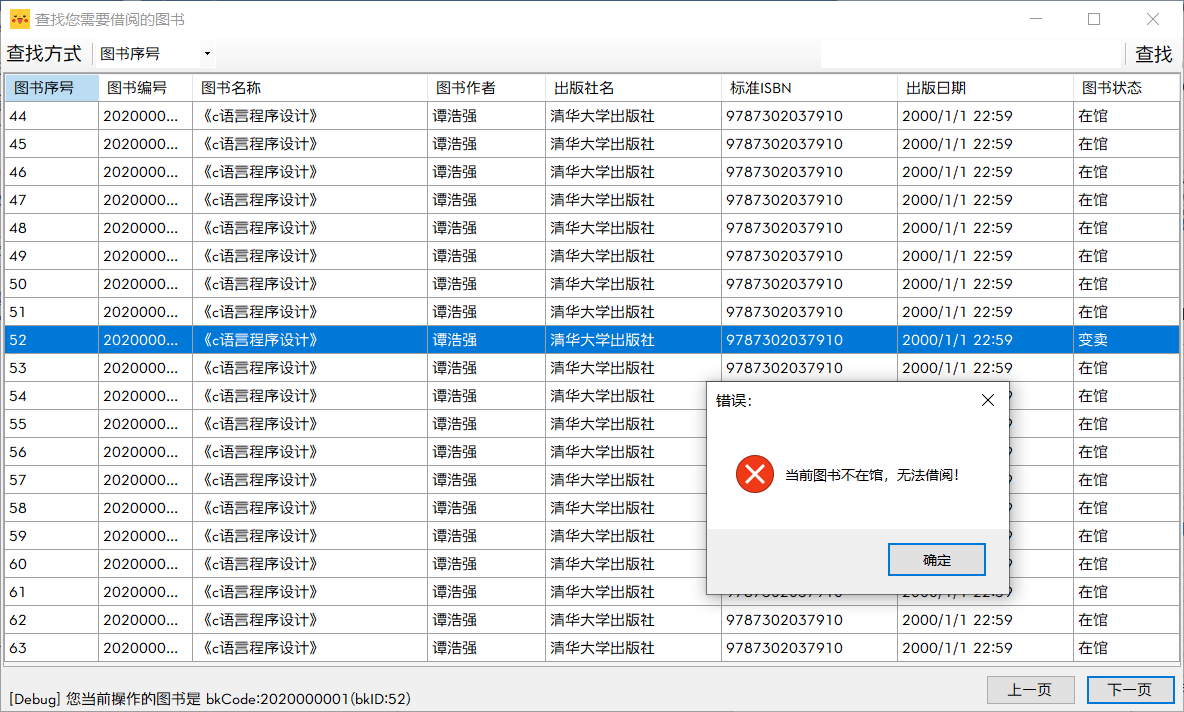


图6.4.1尝试借阅不在馆的图书

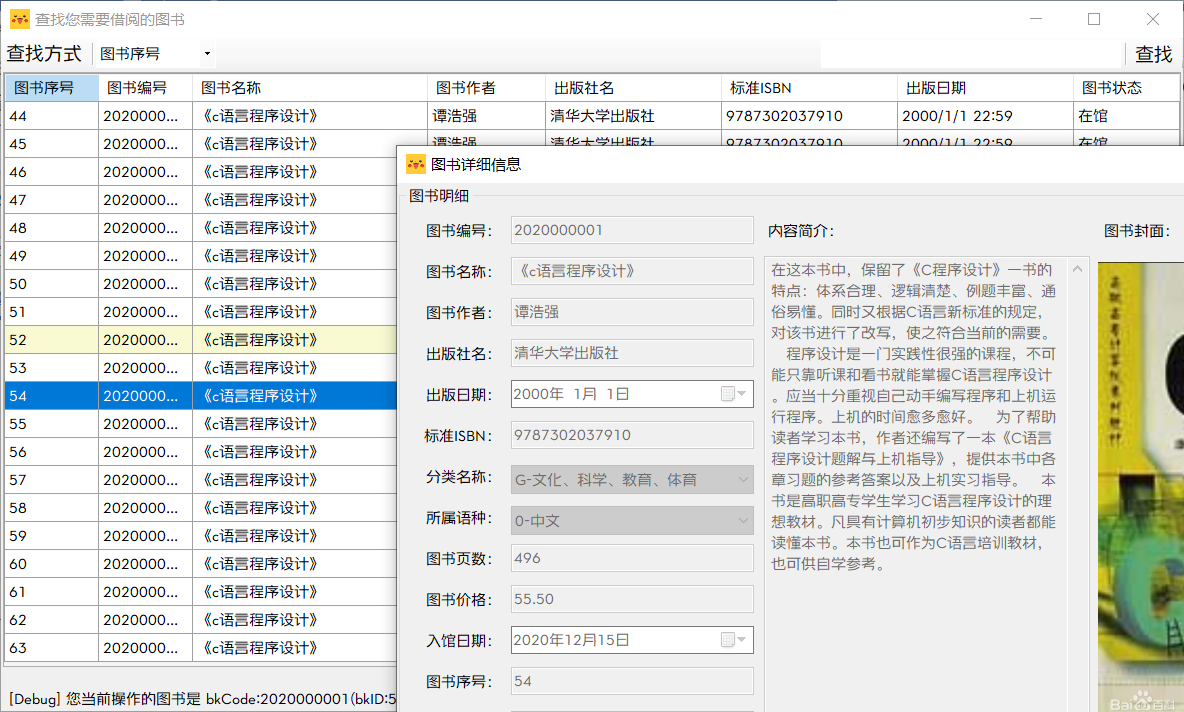


图6.4.2查看图书信息，此时无权修改

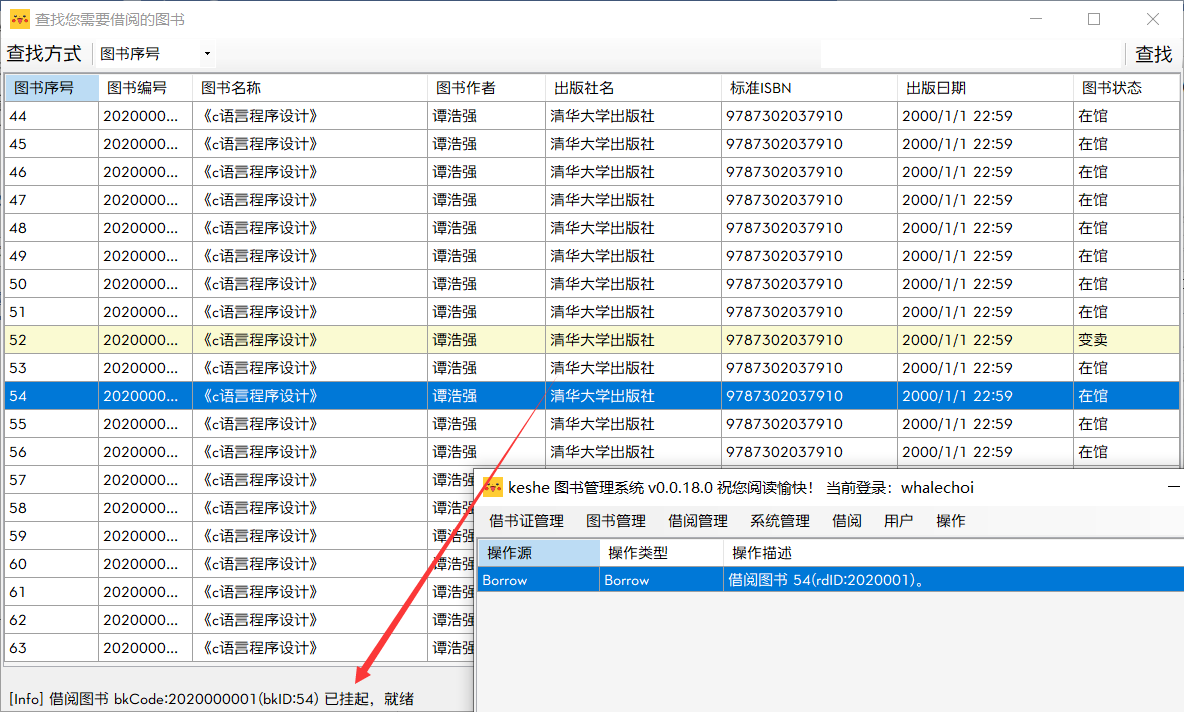


图6.4.3借阅第54号图书操作挂起

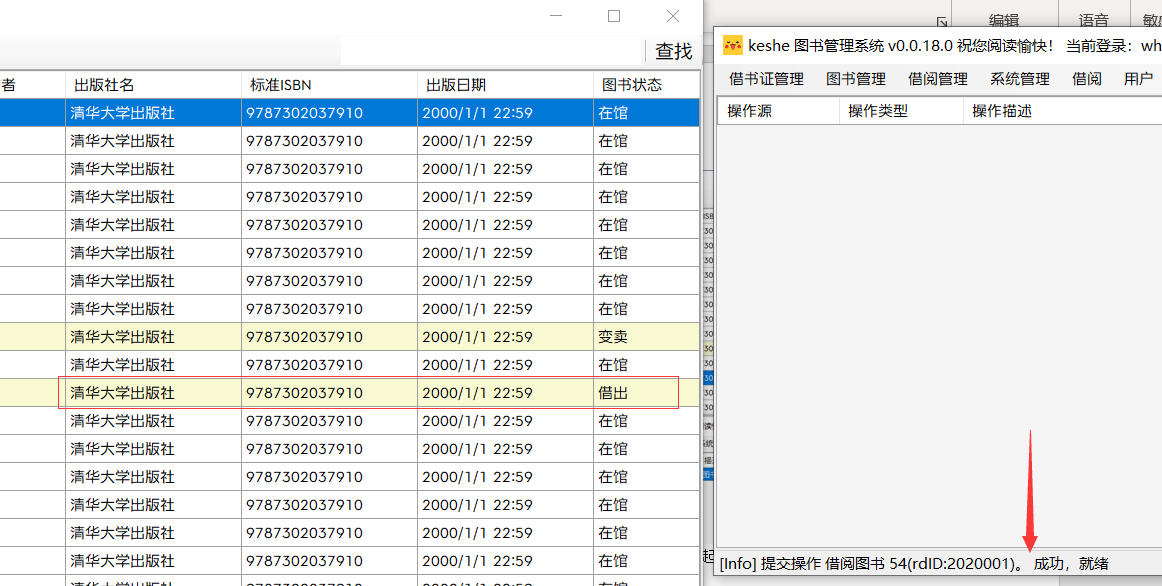


图6.4.4提交操作后借阅成功

### 6.5 续借图书

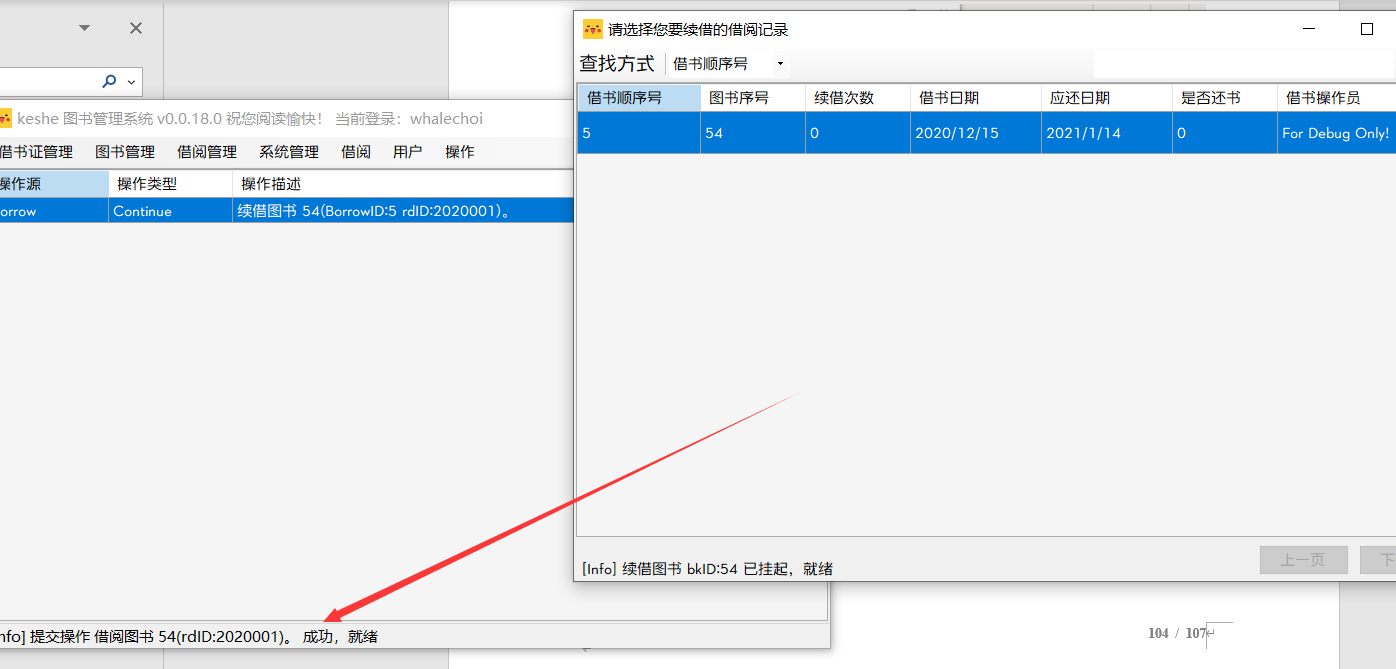


图6.5.1续借操作被挂起

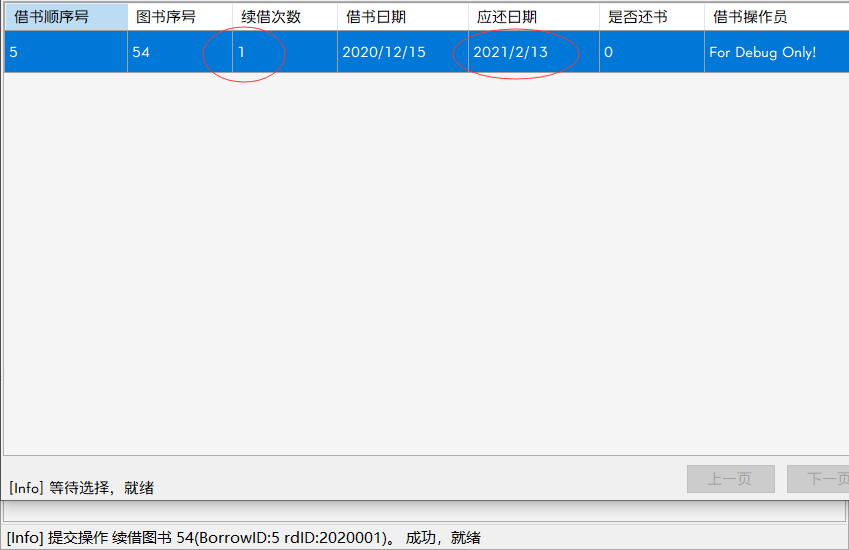


图6.5.2续借成功

### 6.6 归还图书

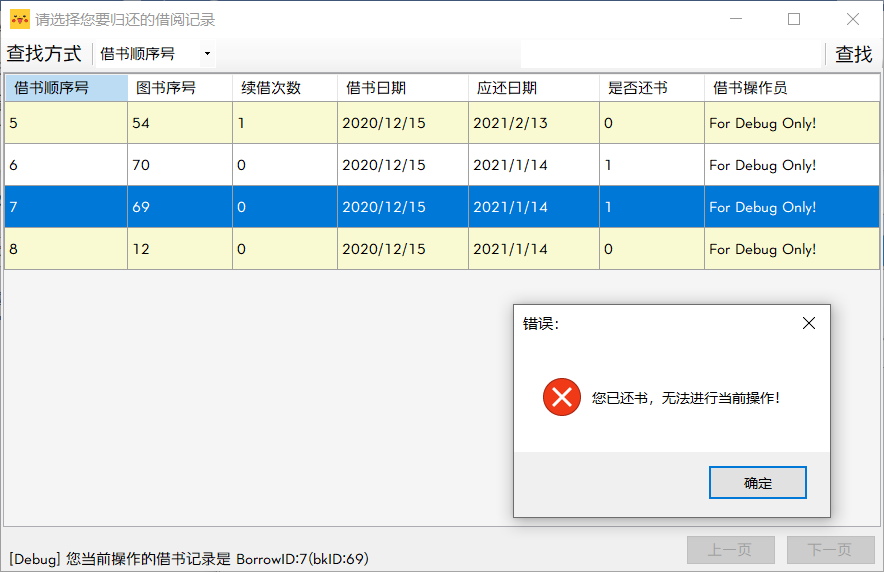


图6.6.1尝试归还已还书的借书记录

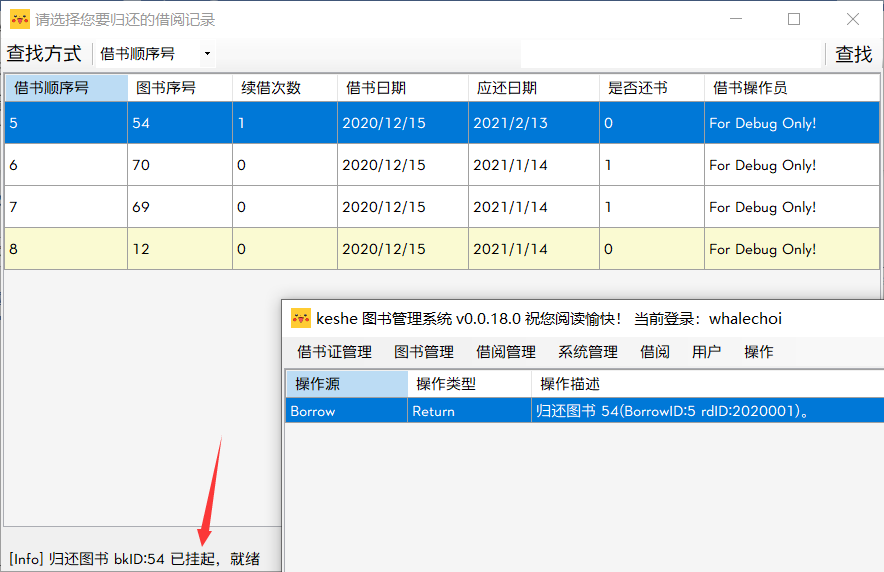


图6.6.2归还图书操作挂起

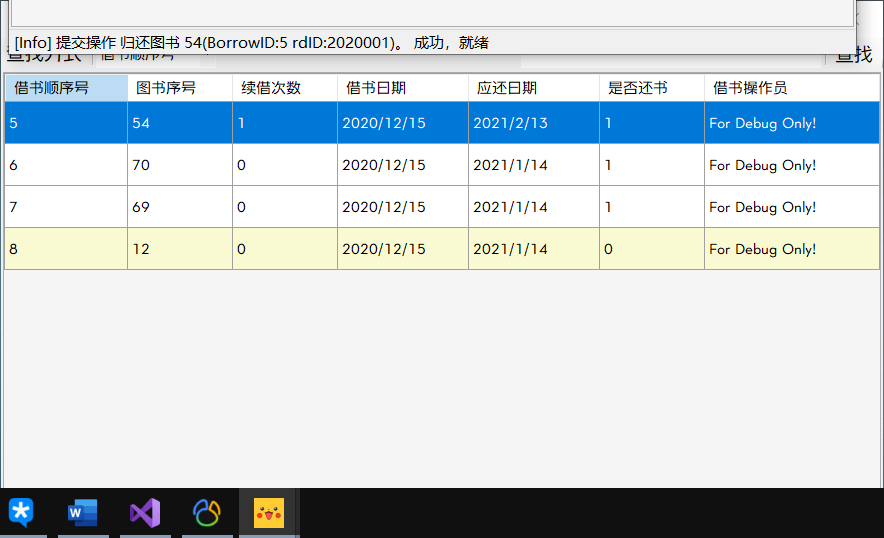


图6.6.3提交操作，归还图书成功

### 6.7 权限管理

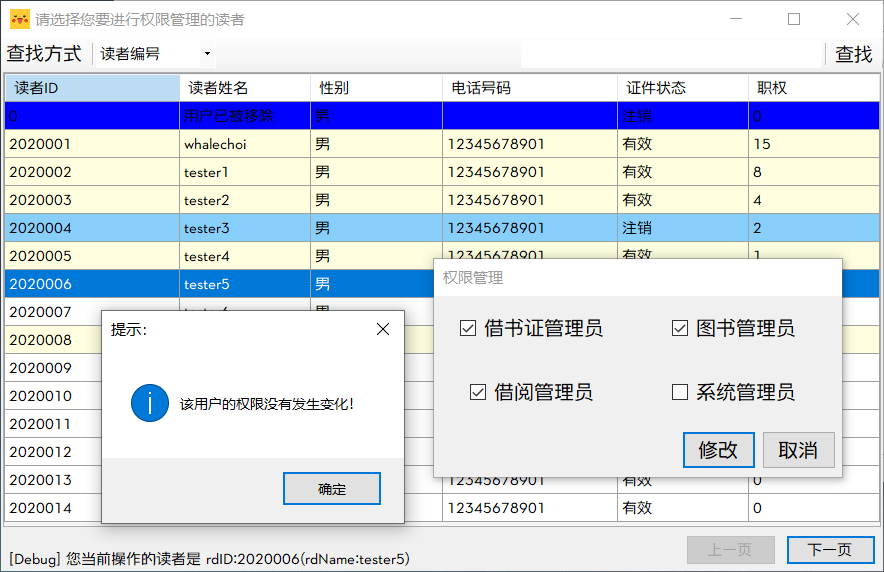


图6.7.1职权没变，修改失败

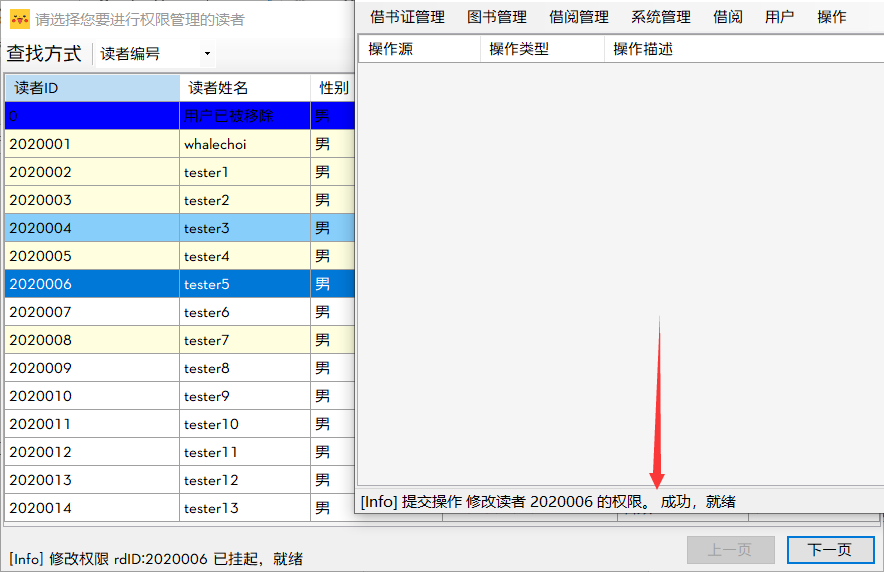


图6.7.2职权改变，提交操作，修改权限成功

### 6.8 密码修改

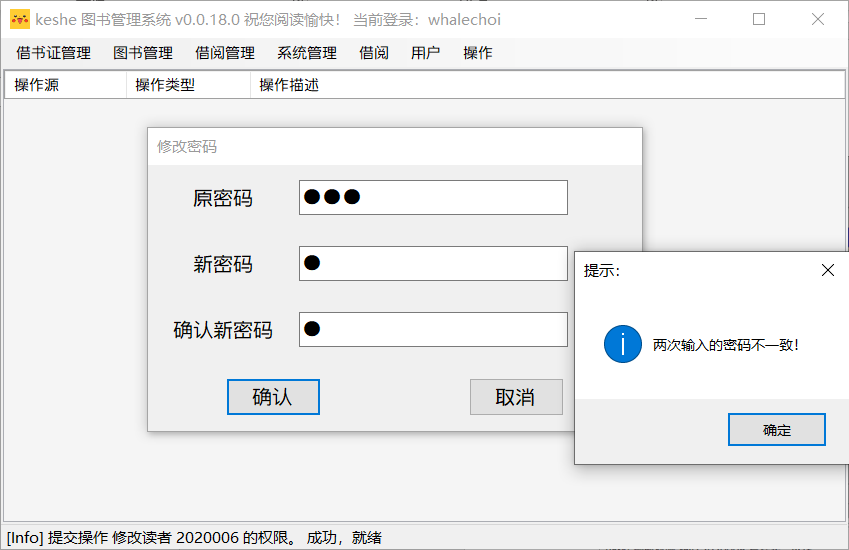


图6.8.1修改密码失败

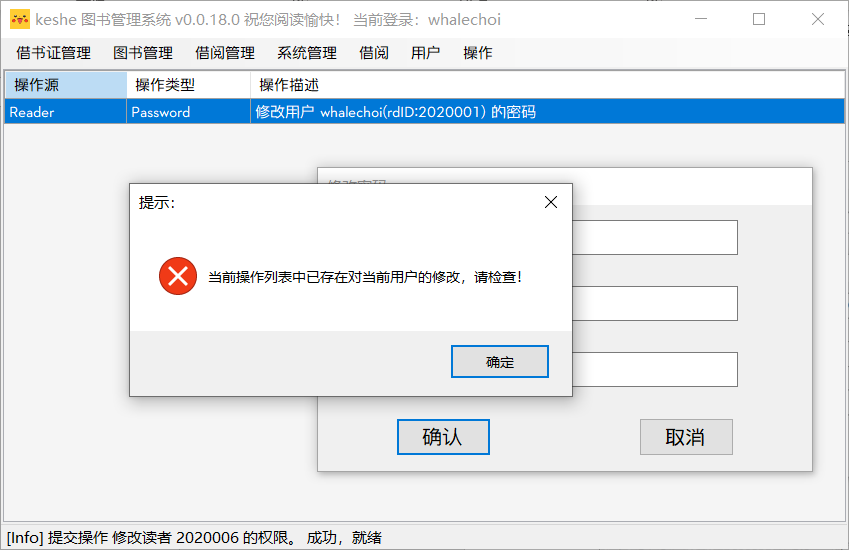


图6.8.2尝试重复修改密码引起程序故障

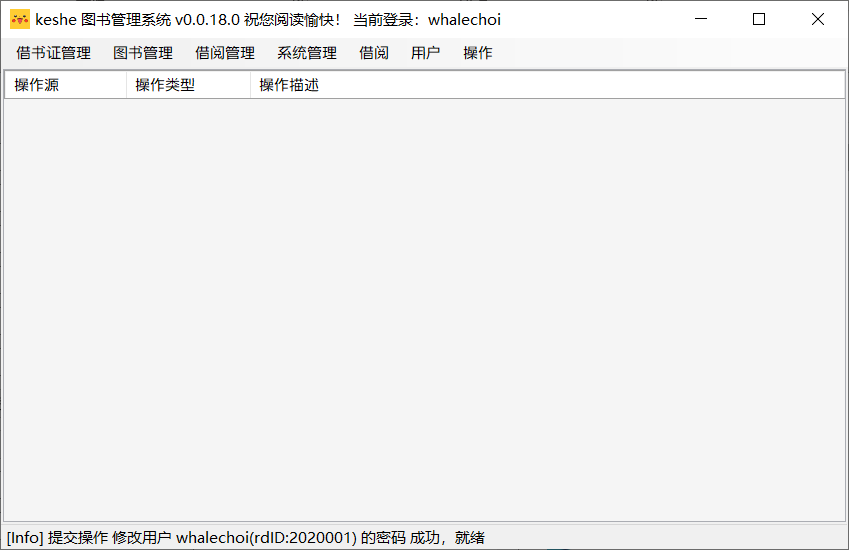


图6.8.3提交操作，密码修改成功

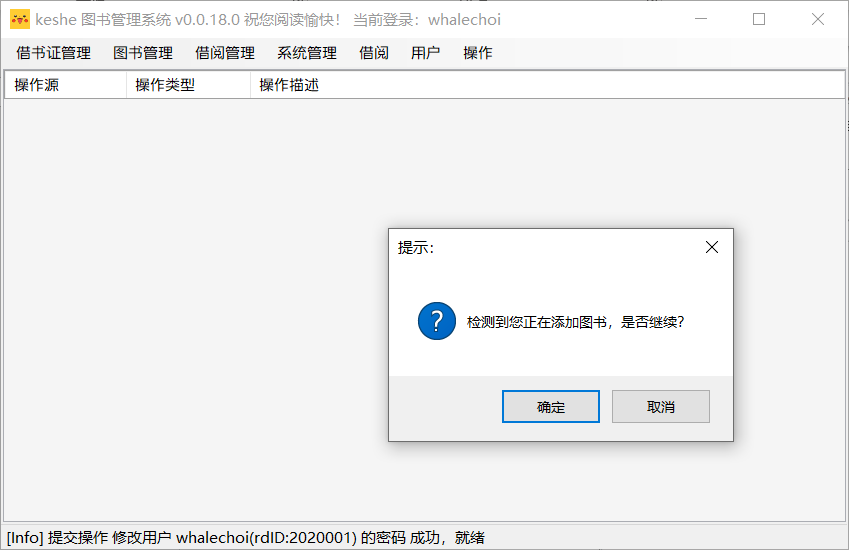


图6.8.4测试结束，尝试未关闭其他窗口时退出系统

## 七、课程设计小结

本次课设，我独立设计了这套比较简易的图书管理系统，编写和调试程序过程中，我遇到了大量的异常去处理，其中有各种奇奇怪怪的问题，比如SQL语句报错，用户一些不恰当的行为导致的软件报错，异常，甚至崩溃；整个程序的核心操作其实并不是最难的部分，最难处理的部分恰恰是这些小细节，在实际使用过程中，总有一些用户进行不恰当的操作，试图破解程序等行为，为此，在程序编写过程中为了安全性，对SQL语句的执行要十分谨慎，杜绝SQL注入；对于用户的密码要加密；对于每个用户的重复操作要进行判断；用户所提供的信息也都要进行初步的代码审核。这让我懂得了软件安全的重要性，在实际项目维护过程的繁琐也要有耐心去一个一个处理，绝不能马虎大意。

指导老师意见：

成绩: 教师签名：

年 月 日