目　录

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

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

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

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

3.1 读者管理

读者管理即借书证管理，需要管理员具有读者管理权限。借书证管理包括：办理借书证、借书证变更、借书证挂失、解除挂失、补办借书证、注销借书证。

借书证（读者）可分为 2 种类别：教师、学生。

借书证（教师）=借书证号、姓名、性别、所在单位、办证日期、照片等。

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

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

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。

3.1.1 办理借书证

管理员给读者办理新借书证。已读者的学号或者教师职工号作为借书证读者编号，根据读者的身份选择相应的读者类别、所在单位以及读者的其他相关信息，信息输入完成即可办证。

3.1.2 变更借书证

管理员给读者变更借书证的相关信息，如读者的类别、有效期、所在单位，以及其他信息等，但借书证号不可更改。

3.1.3 挂失借书证

管理员给读者办理借书证挂失，将读者借书证的证件状态改为挂失，此账号将无法登录和借书。

3.1.4 解除挂失

管理员给读者办理借书证解除挂失，将读者借书证的证件状态改为有效，此账号将可以正常登录和借书。

3.1.5 注销借书证

读者已毕业不在使用该借书证，管理员将注销这些读者的借书证，账号将不能登录和借书。

3.2 图书管理

图书管理需要管理员具有图书管理权限，管理业务包括：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）。

3.2.1 新书入库

图书管理员在新书进入图书馆前，录入新书的相关信息，输入图书的编号，书号，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、入馆日期等。

3.2.2 图书信息维护

图书管理员可以对入库的图书信息进行查删改操作。

3.2.3 在馆图书变卖与销毁处理

图书管理员可以对长期没有借阅记录且失去保存价值的图书进行变卖或销毁处理，并将图书的相关状态修改，读者将无法再借阅变卖和销毁的图书。

3.3 借阅管理

借阅管理需要管理员具有借阅读者管理权限，管理业务包括借书、续借、还书等。还书过程涉及超期罚款、遗失图书罚款等业务规则。

罚款规则：

（1）超期罚款规则 应罚款金额=超期天数*罚款率，罚款率=0.05 元/天，罚款率可能随时间或读者类别而变化；实际罚款金额<=应罚款金额，根据实际情况可以进行减免。

（2）遗失罚款规则 遗失图书应罚款金额=3*图书单价；实际罚款金额在（1 图书单价，3 图书单价）之间。

（3）遗失罚款规则优先于超期罚款规则。

借书记录=借书证号、书号、借书操作员、借书日期、应还日期

续借记录=借书证号、书号、续借操作员、续借日期、应还日期，续借次数

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

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。

3.3.1 借书

借阅管理员给读者办理借书手续，系统查询读者信息、未归还图书信息（含超期），并进行显示。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。系统判断图书是否在馆，若不在馆，则禁止借书。

创建借阅记录对象（借书顺序号由系统自动产生，借书证号和书号为上述输入值，借书操作员=登录用户，借书日期=系统日期，应还日期=系统日期+可借书天数，续借次数=0，还书日期=NULL，还书操作员=NULL），并标记为未归还；修改读者对象的已借书数量+1；修改图书状态为借出；系统记录借阅对象、读者对象、图书对象。

业务规则：(1)借书证状态为挂失、注销者不能借书；(2)借书数量不能超过可借书数量；(3)有超期未归还图书者不能借书。

3.3.2 续借

借阅管理员给读者办理续借手续，系统查询读者信息、未归还图书信息（含超期），并进行显示。系统判断可否续借（续借次数<可续借次数，读者状态为有效）。

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。

3.3.3 还书

借阅管理员给读者办理还书手续，系统查询读者是否借书逾期，处理超期罚款、遗失图书罚款处理两种情况。

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。

3.4 用户管理

每个有效用户都可以登录此图书管理系统，根据用户的身份判断用户具有的权限。

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。如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。

3.4.1 用户登录

用户登录系统，如果用户编号和密码不匹配，用户状态不是有效状态将登录失败。邓丽系统后根据登录用户权限显示或隐藏相应的系统功能（菜单等）。

3.4.2 密码修改

用户登录系统成功后，可以看到用户自己的相关信息，并可以修改自己的登录密码，然后需要重新登录。

3.4.3 用户管理

系统管理员可以新增系统的管理员，对图书馆的工作人员进行权限管理，登录用户为系统管理员，图书馆工作人员也需要办理借书证，也可以借书。

后置条件：修改读者的管理员角色，系统记录读者信息。

四、系统设计

4.1 数据库设计

所用数据库为开源免费的 MySQL 数据库，版本为 5.7，数据库名 bookmanagersystem，数据库引擎为 InnoDB，字符集为 utf8，整体的 ER 图如 4-1-1 图所示。

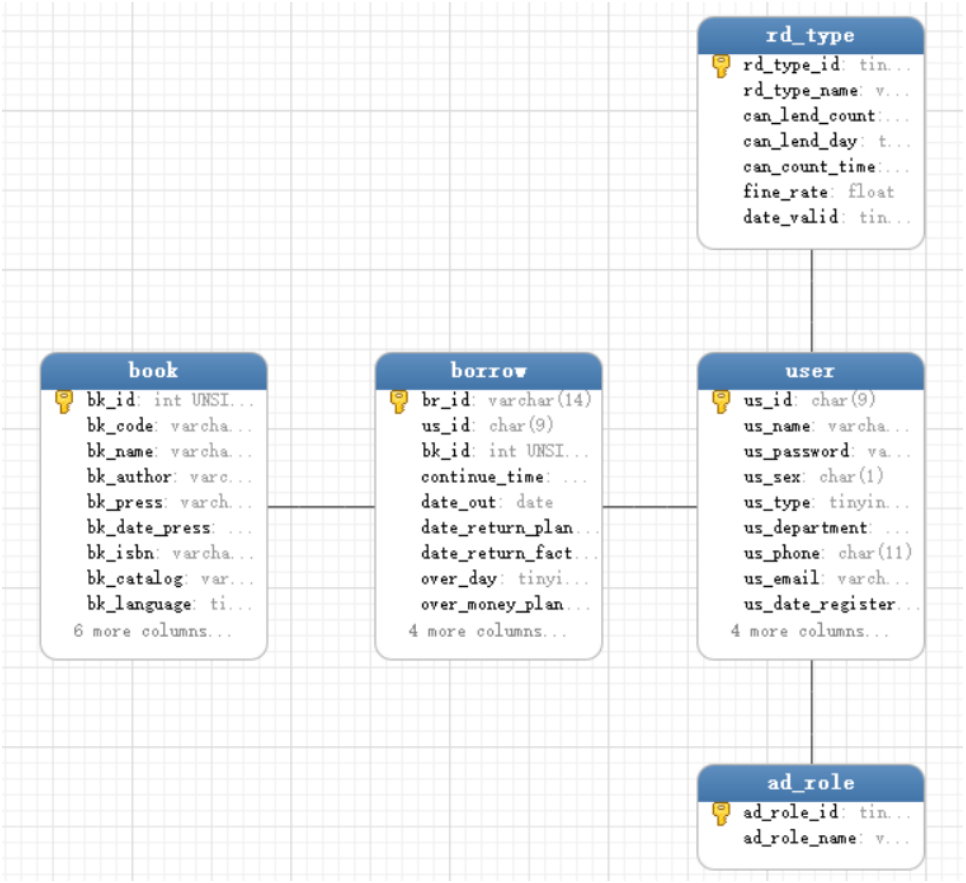

图 4-1-1 数据库 ER 图

读者类型表（rd_type），如表 4-1-1 所示。

表 4-1-1 读者类型表

序号	字段名	数据类型	说明
1	rd_type_id	tinyint(1)	读者类型编号（主键）
2	rd_type_name	varchar(10)	读者类型名称（非空）
3	can_lend_count	tinyint(1)	可借数量（非空）
4	can_lend_day	tinyint(1)	可借天数（非空）
5	can_count_time	tinyint(1)	可借次数（非空）
6	fine_rate	float(3,2)	罚款率（非空）
7	date_valid	tinyint(1)	证件有效期（非空），0 代表永久有效

管理角色表（ad_role）表，如表 4-1-2 所示。

表 4-1-2 管理角色表

序号	字段名	数据类型	说明
1	ad_role_id	tinyint(1)	管理角色编号（主键）
2	ad_role_name	varchar(20)	管理角色名称（非空）

用户信息表（user），如表 4-1-3 所示。

表 4-1-3 用户信息表

序号	字段名	数据类型	说明
1	us_id	char(9)	用户编号（学号，职工号）（主键）
2	us_name	varchar(10)	用户姓名（非空）
3	us_password	varchar(20)	用户密码（非空）默认为 123
4	us_sex	char(1)	用户性别
5	us_type	tinyint(1)	用户类型（外键）
6	us_department	varchar(20)	用户单位
7	us_phone	char(11)	用户电话
8	us_email	varchar(25)	用户邮箱
9	us_date_register	date	注册时间（非空）
10	us_photo	blob	用户头像
11	us_status	char(2)	用户状态（非空）
12	us_borrow_count	tinyint(1)	已借书数量（非空）
13	us_admin_role	tinyint(1)	用户管理角色（外键）

图书信息表（book），如表 4-1-4 所示。

表 4-1-4 图书信息表

序号	字段名	数据类型	说明
1	bk_id	int(11) unsigned	图书序号（主键）
2	bk_code	varchar(20)	图书条码（非空）
3	bk_name	varchar(50)	书名（非空）
4	bk_author	varchar(30)	作者（非空）
5	bk_press	varchar(50)	出版社
6	bk_date_press	date	出版日期
7	bk_isbn	varchar(15)	ISBN 书号
8	bk_catalog	varchar(30)	分类号
9	bk_language	tinyint(1)	语言
10	bk_pages	int(11)	页数
11	bk_price	float	图书价格
12	bk_date_in	date	入馆日期
13	bk_brief	tinytext	图书简介
14	bk_cover	blob	图书封面
15	bk_status	varchar(2)	图书状态（非空）

借阅信息表（borrow），如表 4-1-5 所示。

表 4-1-5 借阅信息表

序号	字段名	数据类型	说明
1	br_id	varchar(14)	借阅记录号（主键）
2	us_id	char(9)	用户编号（外键）
3	bk_id	int(11) unsigned	图书序号（外键）
4	continue_time	tinyint(1) unsigned	续借次数，默认为 0
5	date_out	date	借出日期（非空）
6	date_return_plan	date	理论应归还最晚日期（非空）

7	date_return_fact	date	实际归还日期
8	over_day	tinyint(3)	逾期天数
9	over_money_plan	float	理论罚款
10	over_money_fact	float	实际罚款
11	has_return	char(1)	是否归还
12	operator_lend	varchar(10)	借书操作员
13	operator_return	varchar(10)	还书操作员

4.2 系统架构

简单的来说，本系统使用 MVC 设计模式，采用三层体系结构，M 表示数据模型类，V 表示界面视图，属于表示层（UI），C 表示控制类，包含数据访问层（DAL）和业务逻辑层（BLL），各层的依赖关系如图 4-2-1 所示。

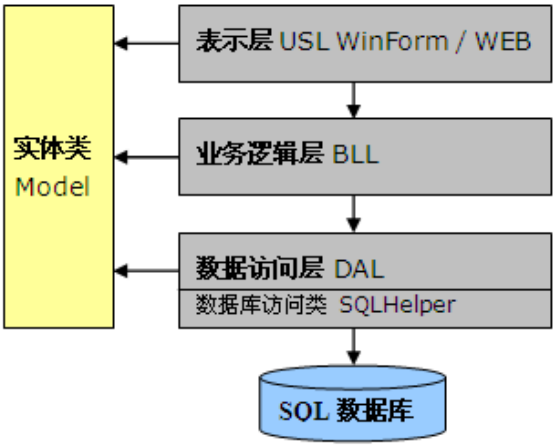

图 4-2-1 三层架构示意图

表示层（USL）：也称 UI，提供交互式界面，形式：Java Swing GUI 或 Web 界面。

业务逻辑层（BLL）：实现业务功能，为表示层提供服务。。

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。

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。

4.3 SSH 框架

本系统后端采用 Java Web 的 SSH 框架，即 Struts2，Spring，Hibernate 三大框架，整个系统的应用架构如图 4-3-1 所示。

Struts2 负责 Web 层，主要是对访问控制进行转发，各类基本参数校验，或者不复用的业务简单处理等。

Spring 是一站式的框架，它将三个框架整合在一起，负责对象的创建配置，减弱各层之间的类依赖关系，从而使整个应用具有“高内聚、低耦合”的特点，便于系统的维护。

Hibernate 是数据持久层的框架，可以以面向对象的方式来操作数据表，负责 DAO 层，与 MySQL 数据库进行交互。

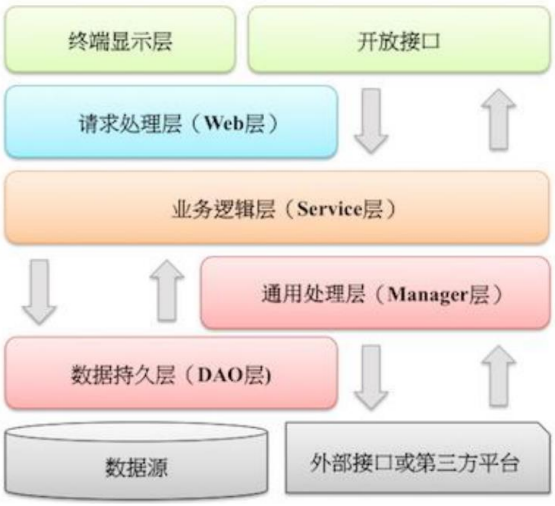

图 4-3-1 应用分层架构图

4.4 类和接口总体设计

整个应用系统后端目录如图 4-4-1 所示，其中 cc.wenshixin.entity 包中为实体类，cc.wenshixin.dao 包中为数据访问类和接口，cc.wenshixin.service 包中为业务逻辑处理类，cc.wenshixin.action 包中为页面访问请求处理类。

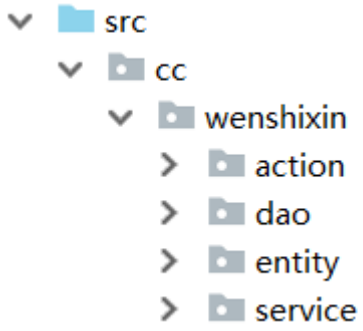

图 4-4-1 系统后端目录图

各个包中的类设计如表 4-4-1 所示。

表 4-4-1 各个包中类设计

包	类名	说明
cc.wenshixin.entity（实体）	AdRole	管理角色实体类
	RdType	读者类型实体类
	User	用户实体类
	Book	图书实体类
	Borrow	借阅记录实体类
cc.wenshixin.dao（数据访问）	BaseDao	基础数据访问接口，定义了基本的增删改查抽象方法，其他的接口都继承该接口。
	BaseDaoImpl	基础数据访问接口实现类，实现了 BaseDao 接口中的抽象方法，使用反射来使其他子类直接使用实现的方法。
	AdRoleDao	管理角色类数据访问接口
	AdRoleDaoImpl	管理角色类数据访问接口实现类

	RdTypeDao	读者类型类数据访问接口
	RdTypeDaoImpl	读者类型类数据访问接口实现类
	UserDao	用户类数据访问接口
	UserDaoImpl	用户类数据访问接口实现类
	BookDao	图书类数据访问接口
	BookDaoImpl	图书类数据访问接口实现类
	BorrowDao	借阅记录数据访问接口
	BorrowDaoImpl	借阅记录数据访问接口实现类
cc.wenshixin.service（业务逻辑处理）	AdRoleService	管理角色业务逻辑类
	RdTypeService	读者类型业务逻辑类
	UserService	用户业务逻辑类
	BookService	图书业务逻辑类
	BorrowService	借阅记录业务逻辑类
cc.wenshixin.action（访问请求处理）	AdRoleAction	管理角色的请求处理类
	RdTypeAction	读者类型的请求处理类
	UserAction	用户的请求处理类
	BookAction	图书的请求处理类
	BorrowAction	借阅记录请求处理类

4.5 Web 页面设计

整个应用系统的前端目录如图 4-5-1 所示，web/index.jsp 为登录系统页面，web/system/index.jsp 为进入系统后台的主页，web/util/下为前端的重用代码段，其他的页面引用了这些代码段，提高了页面的灵活性和重用性，web/system/admin/下为管理员信息的相关页面，其他文件夹也是如此。

图 4-5-1 系统前端目录图

五、系统实现

5.1 SSH 框架配置

将 SSH 框架所需的 jar 包导入到项目中，Spring 的总配置文件 spring.xml，Strusts2 的总配置文件 struts.xml，Hibernate 的总配置文件 hibernate.cfg.xml。在 spring.xml 中引入其他实体类的 spring 的配置文件，在 struts.xml 中引入其他的 struts 的配置文件，hibernate.cfg.xml 引入实体类的映射配置文件，如图 5-1-1 所示。

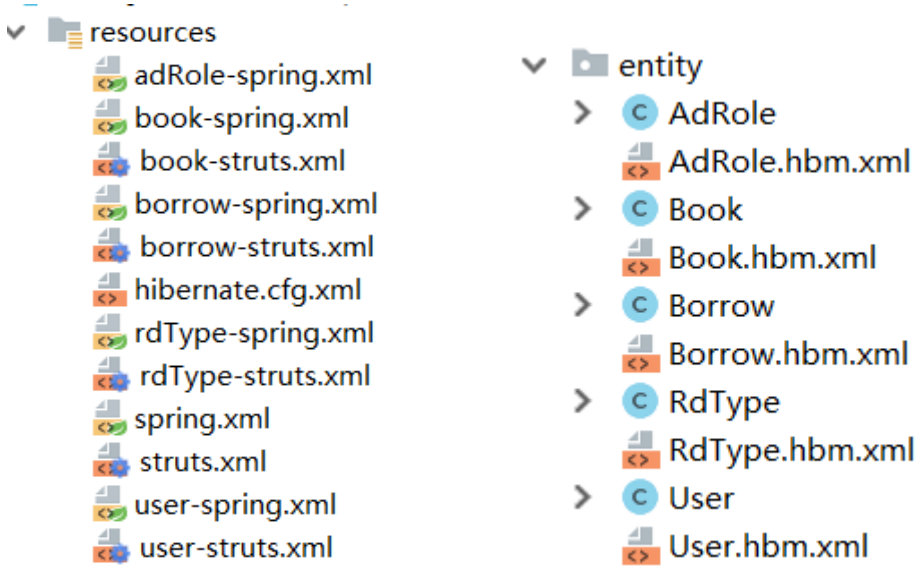

图 5-1-1 SSH 框架的配置文件

spring.xml 文件代码如下

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">
  <!-- 配置数据库信息 -->
  <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <!-- 注入属性值 -->
    <property name="driverClass" value="com.mysql.jdbc.Driver"></property>
    <property name="jdbcUrl"
value="jdbc:mysql://localhost:3306/bookmanagersystem?useSSL=false"></property>
    <property name="user" value="weizhiwen"></property>
    <property name="password" value="123456"></property>
```

```
  </bean>
  <!-- 配置 sessionFactory 创建 -->
  <bean id="sessionFactory"
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
    <!-- 制定数据库的信息 -->
    <property name="dataSource" ref="dataSource"></property>
    <!-- 指定使用的 hibernate 核心配置文件的位置 -->
    <property name="configLocations" value="classpath:hibernate.cfg.xml"></property>
  </bean>
  <!-- 配置事务管理器 -->
  <bean id="transactionManager"
class="org.springframework.orm.hibernate5.HibernateTransactionManager">
    <!-- 注入 sessionFacory 对象 -->
    <property name="sessionFactory" ref="sessionFactory"></property>
  </bean>
  <!-- 开启事务注解 -->
  <tx:annotation-driven transaction-manager="transactionManager"/>
  <!-- 开启注解扫描 -->
  <context:component-scan base-package="cc.wenshixin"></context:component-scan>
  <!-- 引入 spring 对象配置文件 -->
  <import resource="classpath:user-spring.xml"/>
  <import resource="classpath:rdType-spring.xml"/>
  <import resource="classpath:adRole-spring.xml"/>
  <import resource="classpath:book-spring.xml"/>
  <import resource="classpath:borrow-spring.xml"/>
</beans>
```

struts.xml 文件代码如下

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
    <package name="crm" extends="struts-default" namespace="/"></package>
    <!-- 引入其他 struts 配置文件 -->
    <include file="user-struts.xml"></include>
    <include file="rdType-struts.xml"></include>
    <include file="book-struts.xml"></include>
    <include file="borrow-struts.xml"></include>
</struts>
```

hibernate.cfg.xml 文件代码如下

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
```

```
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
      <!-- hibernate 配置 -->
      <property name="show_sql">true</property>
      <property name="format_sql">true</property>
      <property name="hbm2ddl.auto">update</property>
      <property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>
      <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
      <!-- 引入映射文件 -->
      <mapping resource="cc/wenshixin/entity/AdRole.hbm.xml"/>
      <mapping resource="cc/wenshixin/entity/Borrow.hbm.xml"/>
      <mapping resource="cc/wenshixin/entity/Book.hbm.xml"/>
      <mapping resource="cc/wenshixin/entity/RdType.hbm.xml"/>
      <mapping resource="cc/wenshixin/entity/User.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

5.2 entity 类实现

User 实体类代码如下

```
package cc.wenshixin.entity;
import java.sql.Date;
import java.util.HashSet;
import java.util.Set;

public class User {
    private String usId;
    private String usName;
    private String usPassword;
    // 读者类型
    private RdType rdType;
    private String usSex;
    private String usDepartment;
    private String usPhone;
    private String usEmail;
    private Date usDateRegister;
    private byte[] usPhoto;
    private String usStatus;
    private Integer usBorrowCount;
    // 管理角色
    private AdRole adRole;
    // 借阅记录
    private Set<Borrow> setBorrow = new HashSet<Borrow>();
    public User() {}
```

```
    public User(String usId, String usName, String usPassword, String usStatus, AdRole
adRole) {
        this.usId = usId;
        this.usName = usName;
        this.usPassword = usPassword;
        this.usStatus = usStatus;
        this.adRole = adRole;
    }
    // 此处省略 getter 和 setter 方法
}
```

5.3 dao 类和接口实现

基础数据访问接口类 BaseDao

```
package cc.wenshixin.dao;
/**
 * 基础接口类
 * @author 魏志文
 * @param <T> T 为任意类型
 */
import java.util.List;
public interface BaseDao<T> {
    //添加
    void add(T t);
    //修改
    void update(T t);
    //删除
    void delete(T t);
    //根据 id 查询
    T findOne(int id);
    //根据 id 查询
    T findOne(String id);
    //查询所有
    List<T> findAll();
}
```

基础数据访问接口实现类 BaseDaoImpl

```
package cc.wenshixin.dao;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.util.List;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;
public class BaseDaoImpl<T> extends HibernateDaoSupport implements BaseDao<T> {
    private Class pClass;
    public BaseDaoImpl() {
        //1.得到当前运行继承类的 Class
```

```
        Class clazz = this.getClass();
        //2.得到运行类的父类参数化类型 BaseDaoImpl<类名>
        Type type = clazz.getGenericSuperclass();
        ParameterizedType ptype = (ParameterizedType) type;
        //3.得到实际类型参数<类名>里面的类名
        Type[] types = ptype.getActualTypeArguments();
        Class tclass = (Class) types[0];
        this.pClass = tclass;
    }
    @Override
    public void add(T t) {
        this.getHibernateTemplate().save(t);
    }
    @Override
    public void update(T t) {
        this.getHibernateTemplate().update(t);
    }
    @Override
    public void delete(T t) {
        this.getHibernateTemplate().delete(t);
    }
    @Override
    public T findOne(int id) {
        return (T) this.getHibernateTemplate().get(pClass, id);
    }
    @Override
    public T findOne(String id) {
        return (T) this.getHibernateTemplate().get(pClass, id);
    }
    @Override
    public List<T> findAll() {
        //使用Class里面的getSimpleName() 得到类的名称
        return (List<T>)this.getHibernateTemplate().find("FROM"
            +pClass.getSimpleName());
    }
}
```

5.4 service 类实现

借阅记录的 service 类代码如下

```
package cc.wenshixin.service;
import cc.wenshixin.dao.BookDao;
import cc.wenshixin.dao.BorrowDao;
import cc.wenshixin.dao.UserDao;
import cc.wenshixin.entity.Book;
```

```
import cc.wenshixin.entity.Borrow;
import cc.wenshixin.entity.User;
import javax.transaction.Transactional;
import java.util.List;
@Transactional
public class BorrowService {
    // 属性注入
    private BorrowDao borrowDao;
    public void setBorrowDao(BorrowDao borrowDao) {
        this.borrowDao = borrowDao;
    }
    private UserDao userDao;
    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }
    private BookDao bookDao;
    public void setBookDao(BookDao bookDao) {
        this.bookDao = bookDao;
    }
    public void add(Borrow borrow) {
        borrowDao.add(borrow);
    }
    public Borrow findOne(int id) {
        return borrowDao.findOne(id);
    }
    public Borrow findOne(String id) {
        return borrowDao.findOne(id);
    }
    public void update(Borrow borrow) {
        borrowDao.update(borrow);
    }
    public void delete(Borrow b) {
        borrowDao.delete(b);
    }
    public List<Borrow> findAll() {
        return borrowDao.findAll();
    }
    public void lend(Borrow borrow, User user, Book book) {
        // 事务处理
        // 1.记录添加到 borrow 表中
        borrowDao.add(borrow);
        // 2.读者已借书数量加 1
        userDao.update(user);
        // 3.图书状态修改为借出
```

```
        bookDao.update(book);
    }
    public void returnBook(Borrow borrow, User user, Book book) {
        // 事务处理
        // 1.记录修改
        borrowDao.update(borrow);
        // 2.读者已借书数量减 1
        userDao.update(user);
        // 3.图书状态修改为在馆
        bookDao.update(book);
    }
    public List<Borrow> findConditionByUser(Borrow borrow) {
        return borrowDao.findConditionByUser(borrow);
    }
    public List<Borrow> findConditionByBook(Borrow borrow) {
        return borrowDao.findConditionByBook(borrow);
    }
    public List<Borrow> findAllNoReturn() {
        return borrowDao.findAllNoReturn();
    }
}
```

5.5 action 类示例

用户相关的 action 类

```
package cc.wenshixin.action;
import cc.wenshixin.entity.Book;
import cc.wenshixin.entity.Borrow;
import cc.wenshixin.entity.RdType;
import cc.wenshixin.entity.User;
import cc.wenshixin.service.BookService;
import cc.wenshixin.service.BorrowService;
import cc.wenshixin.service.RdTypeService;
import cc.wenshixin.service.UserService;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;
import org.apache.struts2.ServletActionContext;
import javax.servlet.http.HttpServletRequest;
import java.util.Calendar;
import java.sql.Date;
import java.util.List;
public class BorrowAction extends ActionSupport implements ModelDriven<Borrow>{
    // 模型驱动封装
    private Borrow borrow = new Borrow();
    @Override
```

```
public Borrow getModel() {return borrow;}
// 属性注入
private BorrowService borrowService;
public void setBorrowService(BorrowService borrowService) {
    this.borrowService = borrowService;
}
private UserService userService;
public void setUserService(UserService userService) {
    this.userService = userService;
}
private BookService bookService;
 public void setBookService(BookService bookService) {
    this.bookService = bookService;
}
private RdTypeService rdTypeService;
public void setRdTypeService(RdTypeService rdTypeService) {
    this.rdTypeService = rdTypeService;
}
// 转到添加借阅记录的页面
public String toLendPage() {
    return "toLendPage";
}
// 添加借阅记录的方法
public String lend() {
    HttpServletRequest request = ServletActionContext.getRequest();
    // 1. 判断读者可不可以借，即借书数量是否超过读者类型可借数量
    String usId = borrow.getUser().getUsId();
    User user = userService.findOne(usId);
    int hasCount = user.getUsBorrowCount(); // 读者已借书数量
    RdType rdType = user.getRdType();
    int canCount = rdType.getCanLendCount(); // 读者类型可借书最大数量
    if (hasCount > canCount) {
         request.getSession().setAttribute("errormsg", "已超过读者可借书的最大数量!
     ");
       return "lendFail";
    }
    // 2. 判断图书可不可以被借，即图书状态是否为在馆
    int bkId = borrow.getBook().getBkId();
    Book book = bookService.findOne(bkId);
    String bookStatus = book.getBkStatus();
    if (!"在馆".equals(bookStatus)) {
        request.getSession().setAttribute("errormsg", "图书不在馆中！");
        return "lendFail";
    }
```

```
    // 3. 填写相应的字段，来插入到数据库中
    java.sql.Date dateOut = borrow.getDateOut();
    int days = rdType.getCanLendDay();
    java.sql.Date dateReturn = dateAddSomeDay(dateOut, days);
    borrow.setDateReturnPlan(dateReturn);
    borrow.setOverDay(0);
    borrow.setHasReturn("否");
    borrow.setContinueTime((byte) 0);
    user.setUsBorrowCount(hasCount+1); // 读者的已借书数量加 1
    book.setBkStatus("借出");
    borrowService.lend(borrow,user,book);
    return "lendSuccess";
}
// 根据用户查询的方法
public String userCondition() {
    if(!"".equals(borrow.getUser().getUsId())&& borrow.getUser().getUsId() != null) {
        List<Borrow>borrowList= borrowService.findConditionByUser(borrow);
        HttpServletRequest request = ServletActionContext.getRequest();
        request.setAttribute("borrowList", borrowList);
    } else {
        listNoReturn();
    }
    return "userCondition";
}
// 根据图书查询的方法
 public String bookCondition() {
     if(!"".equals(borrow.getBook().getBkName())&&borrow.getBook().getBkName()!=
 null) {
        List<Borrow>borrowList= borrowService.findConditionByBook(borrow);
        HttpServletRequest request = ServletActionContext.getRequest();
        request.setAttribute("borrowList", borrowList);
    } else {
        listNoReturn();
    }
    return "bookCondition";
}
// 续借的方法
public String continueLend() {
    HttpServletRequest request = ServletActionContext.getRequest();
    String brId = borrow.getBrId();
    Borrow br = borrowService.findOne(brId);
    // 1.判断读者可不可以续借，续借的次数有没有超，所借的天数有没有逾期
    String usId = br.getUser().getUsId();
    User user = userService.findOne(usId);
```

```
    RdType rdType = user.getRdType();
    int continueTime = rdType.getCanCountTime();
    int hasCountTime = br.getContinueTime();
    if(hasCountTime >= continueTime){
          request.getSession().setAttribute("errormsg", "读者续借本书次数已达上限！
          ");
        return "continueLendFail";
    }
          java.sql.Date dateOut = br.getDateOut();
          java.sql.Date returnFact=new java.sql.Date(System.currentTimeMillis()); //
          获取当前的日期
    int betweenDays = (int) dateDifference(dateOut, returnFact);
    int days = rdType.getCanLendDay();
    if (betweenDays > days) {
         request.getSession().setAttribute("errormsg", "读者借阅本书已逾期！");
        return "continueLendFail";
    }
     java.sql.Date dateReturn = dateAddSomeDay(dateOut, days); // 计算应还书的最晚日
     期
    br.setContinueTime((byte) (hasCountTime+1));
    br.setDateOut(returnFact);
    br.setDateReturnPlan(dateReturn);
    borrowService.update(br);
    return "continueLendSuccess";
}
// 转到还书的页面
public String toReturnBookPage() {
    String brId = borrow.getBrId();
    Borrow br = borrowService.findOne(brId);
    String usId = br.getUser().getUsId();
    int bkId = br.getBook().getBkId();
    User user = userService.findOne(usId);
    Book book = bookService.findOne(bkId);
    float bkPrice = book.getBkPrice();
    RdType rdType = user.getRdType();
    int days = rdType.getCanLendDay();
    float rate = rdType.getFineRate();
    java.sql.Date dateOut = br.getDateOut();
     java.sql.Date returnFact= new java.sql.Date(System.currentTimeMillis()); // 获
     取当前的日期
    int betweenDays = (int) dateDifference(dateOut, returnFact);
    if (betweenDays > days) {
        int overDays = betweenDays - days;
        br.setOverDay(overDays);
```

```
            float overMoneyPlan = overDays * rate;
            br.setOverMoneyPlan(overMoneyPlan);
        }
        br.setDateReturnFact(returnFact);
        HttpServletRequest request = ServletActionContext.getRequest();
        request.setAttribute("borrow",br);
        request.setAttribute("price",bkPrice);
        return "toReturnBookPage";
    }
    // 读者还书的方法
    public String returnBook() {
        String usId = borrow.getUser().getUsId();
        int bkId = borrow.getBook().getBkId();
        User user = userService.findOne(usId);
        Book book = bookService.findOne(bkId);
        // 1.读者已借书的数量减 1
        int hasLendCount = user.getUsBorrowCount();
        user.setUsBorrowCount(hasLendCount-1);
        // 2.图书的状态设为在馆状态
        book.setBkStatus("在馆");
        // 3.更新数据表
        borrowService.returnBook(borrow, user, book);
        return "returnBook";
    }
    // 续借和还书页面
    public String continueAndReturn() {
        List<Borrow> borrowList = borrowService.findAllNoReturn();
        HttpServletRequest request = ServletActionContext.getRequest();
        request.setAttribute("borrowList", borrowList);
        return "continueAndReturn";
    }
    // 全部借阅记录
    public String list() {
        List<Borrow> borrowList = borrowService.findAll();
        HttpServletRequest request = ServletActionContext.getRequest();
        request.setAttribute("borrowList", borrowList);
        return "list";
    }
    public String listNoReturn() {
        List<Borrow> borrowList = borrowService.findAllNoReturn();
        HttpServletRequest request = ServletActionContext.getRequest();
        request.setAttribute("borrowList", borrowList);
        return "listNoReturn";
    }
```

```
    // 日期加几天的计算方法
    public java.sql.Date dateAddSomeDay(java.sql.Date sqlDate, int days){
        // java.sql.Date 转 java.util.Date，java.util.Date 再转 java.sql.Date 返回
        java.util.Date utilDate = sqlDate;
        Calendar rightNow = Calendar.getInstance();
        rightNow.setTime(utilDate);
        rightNow.add(Calendar.DAY_OF_YEAR,days);
        java.util.Date tempDate = rightNow.getTime();
        java.sql.Date returnDate = new java.sql.Date(tempDate.getTime());
        return returnDate;
    }
    // 日期相减的计算方法
    public long dateDifference(java.sql.Date sqlDate1, java.sql.Date sqlDate2) {
        // TODO: 2017/12/17
        java.util.Date utilDate1 = sqlDate1;
        java.util.Date utilDate2 = sqlDate2;
        long betweenDays = (utilDate2.getTime() - utilDate1.getTime()) / (1000 * 60 * 60
        * 24);
        return betweenDays;
    }
}
```

5.6 前端界面展示

登录界面，如图 5-6-1 所示。

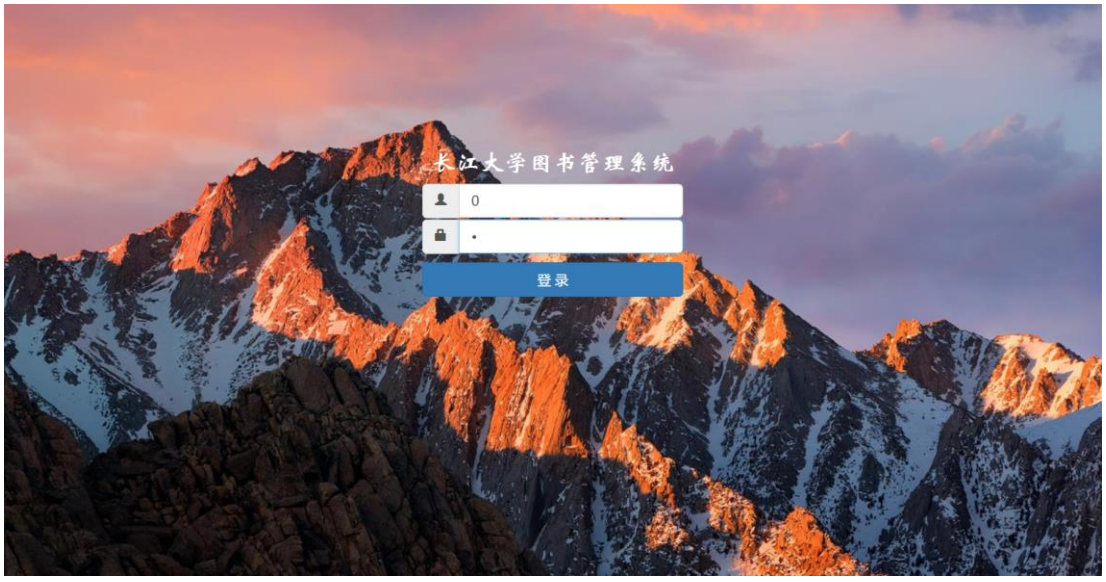

图 5-6-1 登录界面

系统主页面，如图 5-6-2 所示。

图 5-6-2 系统主页

个人信息页面，如图 5-6-3 所示。

图 5-6-3 个人信息

修改密码页面，如图 5-6-4 所示。

图 5-6-4 修改密码

办理借书证页面，如图 5-6-5 所示。

图 5-6-5 办理借书证

借书证信息变更页面，如图 5-6-6 所示。

图 5-6-6 借书证信息变更

新书入库页面，如图 5-6-7 所示。

图 5-6-7 新书入库

图书信息维护页面，如图 5-6-8 所示。

图 5-6-8 图书信息维护

借书页面，如图 5-6-9 所示。

图 5-6-9 借书

续借页面，如图 5-6-10 所示。

图 5-6-10 续借

还书页面，如图 5-6-11 所示。

图 5-6-11 还书

新增管理和查看管理页面和前面的读者办理借书证和管理借书证信息页面相似，这里就不再贴出。

成绩：_____　　教师签名：_____

年　　月　　日