数据库项目开发总结报告

组长：于全坤/2017141411008

组员：赵志远/2017141653005

组员：杨佳瑞/2017141471139

目录

[一．项目概述 4](#_Toc7856)

[1.1概述： 4](#_Toc9319)

[1.2分工： 4](#_Toc23544)

[二．项目背景 5](#_Toc12030)

[2.1 系统总体功能结构 5](#_Toc10109)

[2.1.1管理员功能： 5](#_Toc23294)

[2.1.2读者功能： 6](#_Toc21616)

[2.2 目的与意义 6](#_Toc11113)

[三．技术路线 7](#_Toc7939)

[3.1 主要技术（SSM） 7](#_Toc21332)

[3.1.1 Spring/SpringMVC框架 7](#_Toc24315)

[3.1.2 Mybatis框架 7](#_Toc104)

3.2主要技术及难点.................................................................................................................8

[3.2.1 关键技术 8](#_Toc21608)

[3.2.2 技术难点 9](#_Toc24382)

[四、概要设计 10](#_Toc23334)

[4.1 数据库设计（ER图） 10](#_Toc16227)

[4.2 功能设计 10](#_Toc17217)

[五、详细设计 12](#_Toc15510)

[5.1 数据库设计 12](#_Toc15803)

[5.2 功能设计 14](#_Toc16141)

[5.3 UI设计 26](#_Toc2118)

[六、实现的主要功能 27](#_Toc5620)

[七．总结 27](#_Toc16797)

[7.1亮点 27](#_Toc16850)

[7.2需要完善之处 28](#_Toc4902)

一．项目概述

# 1.1概述：

# 图书管理系统是一个经典的数据库开发案例。整个系统的用户分为两类，一个是图书管理员，另一个则是读者。两种不同的用户有对数据库的不同操作权限，比如用户只有对图书表查权限，而没有更改权限；而管理员则具有查和更改的权限。总体来看，项目的设计思路清晰，逻辑比较简单，易于设计。

# 1.2分工：

杨佳瑞同学主要负责数据库的逻辑设计与物理实现。具体来说，我们在项目之初对整个项目中用到的功能和具体的功能对象进行了分析，然后杨佳瑞同学根据分析结果设计E-R图，再对E-R图完善以后开始进行数据表的设计，而在项目中我们共设计了8张表。其次是数据库的物理实现，杨同学使用python从相关的网站上爬取了大量的书籍信息，并且按照表结构进行信息的分类存放，利用.csv文件的特性，直接用navicat导入本地的数据库的各个表中，实现了本地数据库的物理存放，另外利用mybatis完成数据库连接时的配置.xml文件的代码书写也是由杨同学所完成的，可以说是身兼多职，贡献很大。

赵志远同学则主要负责前端代码的书写以及页面的美化工作，即视图层的实现。视图层涉及了大量的html知识，并且需要对javascirpt的ajax和表单提交数据的方法等知识有着良好的运用能力，所以编码工作比较复杂。由于我们使用了SSM框架，所以前端存粹使用了.jsp来对视图层进行控制，由于每一个不同页面的显示都需要一个不同的.jsp文件来进行控制，所以整体工作量比较大。

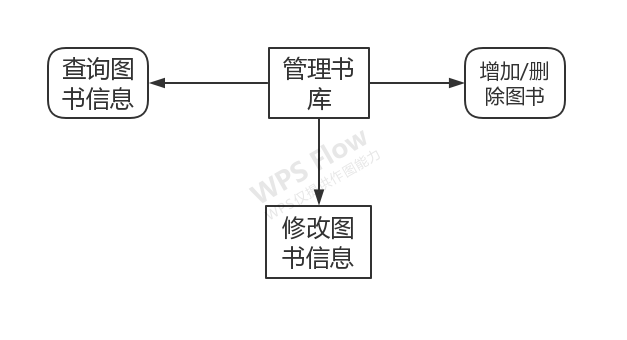
于全坤同学则主要负责业务层代码的书写，由于于同学对java语言比较熟悉，而且对于MVC框架开发有一定的经验，所以对SSM框架的使用能够很快的上手，从而使得项目能够在短短的两周开发时间中，实现动态的运行与交互。

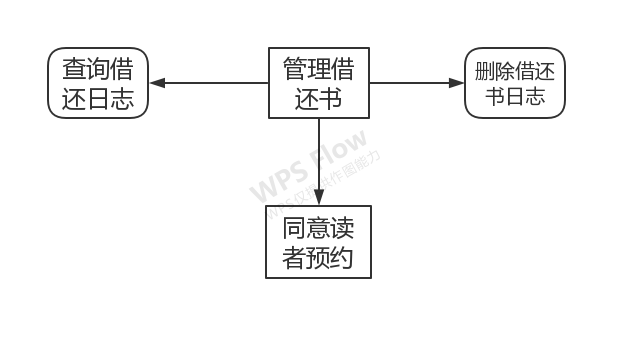
总的来说，三个人的分工虽然各有侧重，但是在编码过程中实际上密不可分，因为后端的结构决定了控制层逻辑，而控制层的返回值又决定了前端的界面的信息，所以在编码过程中，我们三个人互相帮助，共同完成了整个项目的开发。

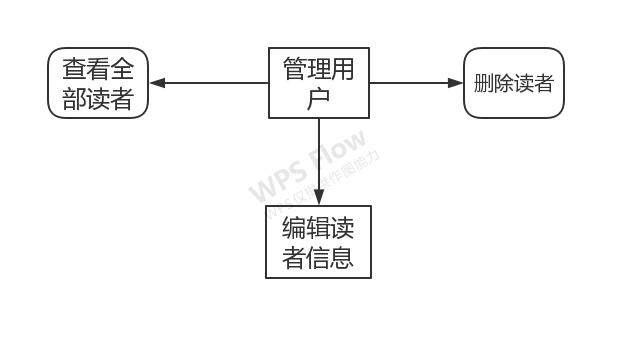
# 二．项目背景

## 2.1 系统总体功能结构

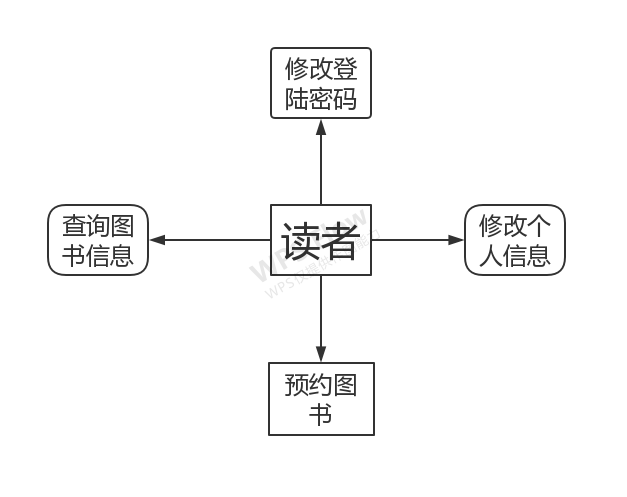
**2.1.1管理员功能：**







**2.1.2读者功能：**



## 2.2 目的与意义

在数据库课程中我们学到了如何针对现实中的某些活动抽象出对应的E-R图，然后利用E-R图设计出对应的表，而在java课程中学习到了java强大的代码移植功能与在网页方面的广泛应用，在前端设计课程中则学到了如何进行静态网页的设计，所以选择图书管理系统这个相对简易的题目进行不同知识的融会贯通。

2.3创新点

主要创新点在于实现了动态预约功能，而不允许读者自行借书的功能。具体来说指的是，用户对某一本书发起预约请求时并不会立即借到该书，而是直到管理员在预约界面同意该用户发起的预约请求以后读者方能得到此书的借阅权限，而一本书的所有副本及本身全部被借出后，读者依旧可以进行预约但是管理员无法同意该预约请求，因为书库中已没有可借的图书了。

# 三．技术路线

## 3.1 主要技术（SSM）

### 3.1.1 Spring/SpringMVC框架

Spring就像是整个项目中装配bean的大工厂，在配置文件中可以指定使用特定的参数去调用实体类的构造方法来实例化对象。也可以称之为项目中的粘合剂。

SpringMVC在项目中拦截用户请求，它的核心Servlet即DispatcherServlet承担中介或是前台这样的职责，将用户请求通过HandlerMapping去匹配Controller，Controller就是具体对应请求所执行的操作。

### 3.1.2 Mybatis框架

mybatis是对jdbc的封装，它让数据库底层操作变的透明。mybatis的操作都是围绕一个sqlSessionFactory实例展开的。mybatis通过配置文件关联到各实体类的Mapper文件，Mapper文件中配置了每个类对数据库所需进行的sql语句映射。在每次与数据库交互时，通过sqlSessionFactory拿到一个sqlSession，再执行sql命令。

3.1.3配合机制

页面发送请求给控制器，控制器调用业务层处理逻辑，逻辑层向持久层发送请求，持久层与数据库交互，后将结果返回给业务层，业务层将处理逻辑发送给控制器，控制器再调用视图展现数据。

3.1.4 开发平台

开发平台为IntelliJ IDEA 2019.3专家版，使用平台中自带的Maven作为项目管理器。项目结构如下：

3.1.5数据库

我们使用了mysql workbench这一图形化操作界面，在本地建立MySQL数据库。同时使用Mybatis对本地的数据库进行连接管理，以及前后端的交互。

3.1.6服务器

服务器我们选择的是比Tomcat更为轻量级的jetty，这使得项目运行所占开销更小。

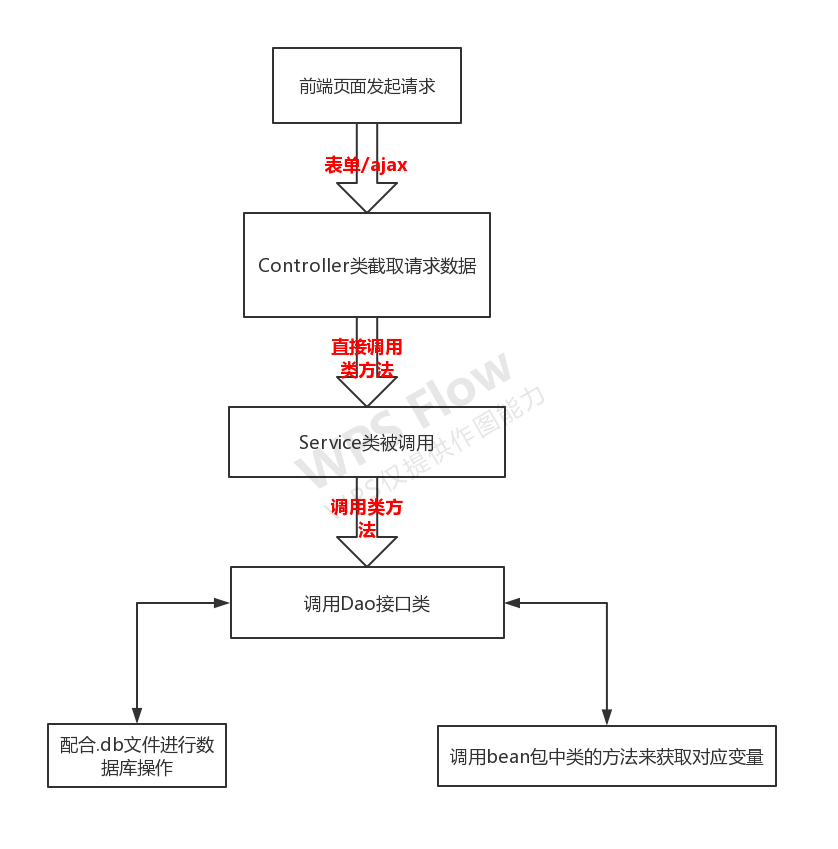
由于Tomcat除了遵循Java Servlet规范之外，自身还扩展了大量JEE特性以满足企业级应用的需求，所以Tomcat是较重量级的，而且配置较Jetty亦复杂许多。但对于大量普通互联网应用而言，并不需要用到Tomcat其他高级特性，所以在这种情况下，使用Tomcat是很浪费资源的。这种劣势放在分布式环境下，更是明显。换成Jetty，每个应用服务器省下那几兆内存，对于大的分布式环境则是节省大量资源。而且，Jetty的轻量级也使其在处理高并发细粒度请求的场景下显得更快速高效。

#### **3.2 关键技术及难点**

### 3.2.1 关键技术

Spring框架是整个项目的粘合剂，它规定了在动态网页编写过程中前后端的交互的编程规范。以我们的项目为例，Spring框架规定了如何对一个页面发送来的表单请求进行捕捉和执行，因为框架中包含了Seivice类和Controller类，Controller类中的@RequestMapping对页面的请求进行捕捉，然后执行Service类中对应的方法，而Service中的方法又是Dao接口提供的方法，Dao接口中的方法一方面可以调用Mybatis下的.db文件进行相应的数据库操作，一方面又需要调用bean类中规定的某个对象的私有变量进行匹配。

整个系统运行结构如下：



### 3.2.2 技术难点

在前后端的业务逻辑层的实现过程中，遇到了很多难题。最开始对整个框架的运行逻辑比较陌生，即使有了部分MVC的动态网页开发基础，但是还是在理解SSM框架的过程时花费了大量的时间。整个SSM框架从前端的请求发起到后端的根据请求对数据库进行更改或者返回符合条件的查询值都有一套完整的规范，并且对@RequestMapping，@Autowired，@Service的运用，都需要大量的经验与学习，所以在做有效的代码编写之前，难点主要在于逻辑理解方面。一旦对整个过程和语法熟悉了以后，进行开发便十分简便且高效了。

难点之二在于jsp文件的书写，由于前端的同学并未对.jsp文件的编码格式有事先了解，所以在开发前端的网页的时候花费了很大的功夫去.jsp文件的编码规范。

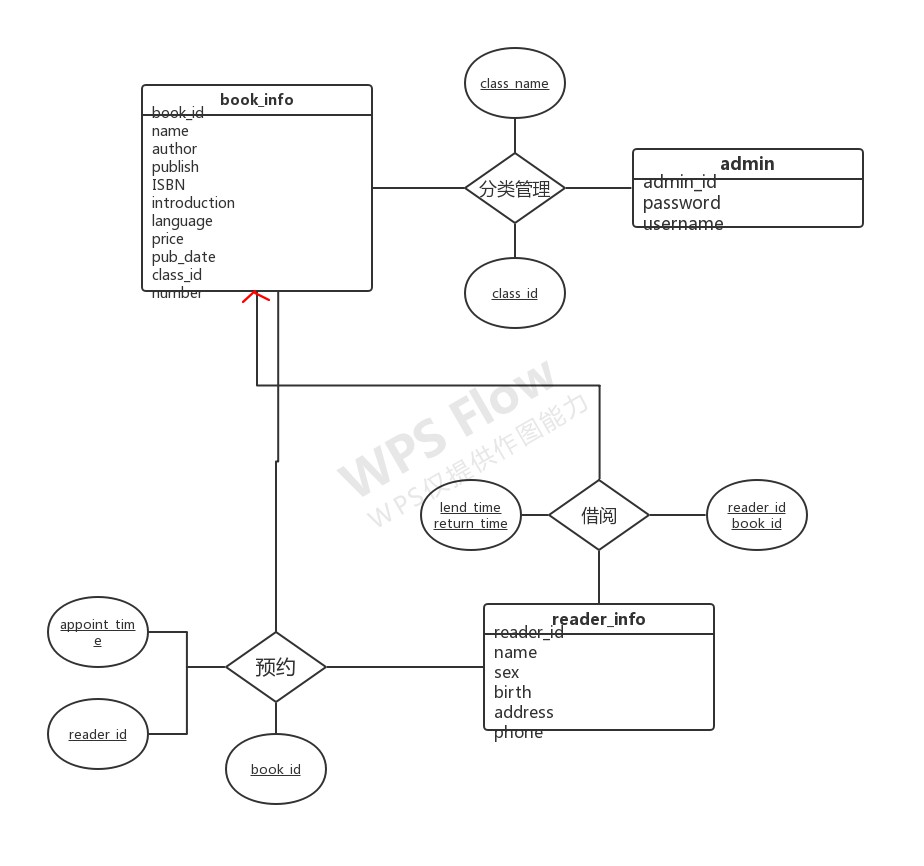
# 四、概要设计

## 4.1 数据库设计（ER图）

数据库设计如下：

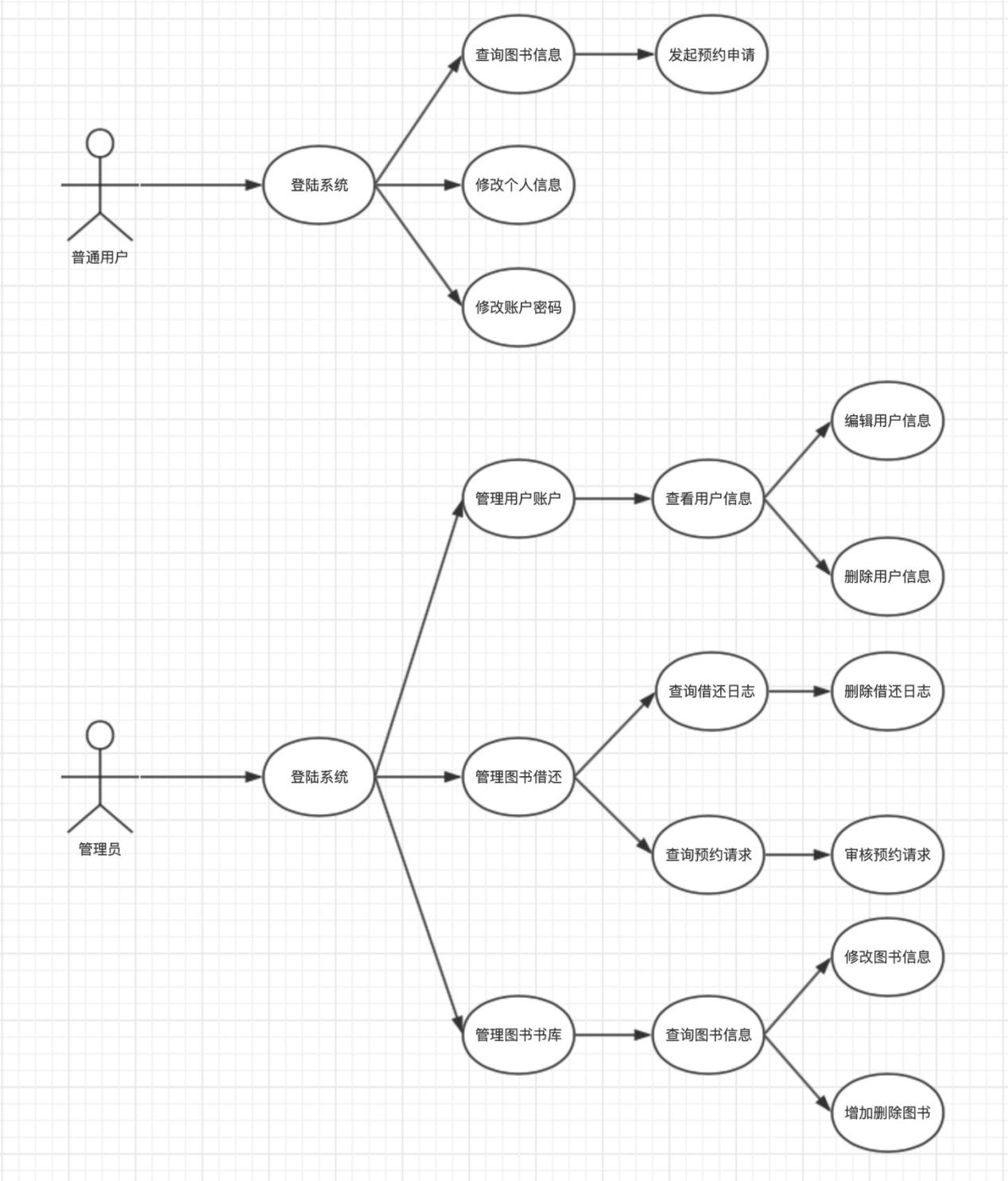
实体对象有三个，分别是Admin、reader、book。

联系有三个，分别是预约，借阅，分类管理。值得注意的时，这三个联系也均有各自的属性，比如预约联系拥有的独有属性是预约时间，借阅联系拥有的独有属性是预约时间和还回时间，分类管理拥有的特有属性是分类号和类名。



## 4.2 功能设计

功能设计用例图如下：



# 五、详细设计

### 5.1 数据库设计

5.1.1逻辑设计

我们的项目总共设计了8张表，分别是book\_info，admin，admin\_info，reader\_card，reader\_info，lend\_list，appointlist，class\_info。

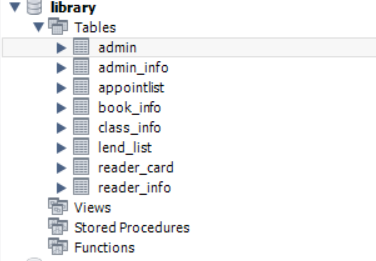
表的具体的逻辑结构如下：

*#### 1. 图书书目表book\_info*| 名 | 类型 | 长度 | 小数点 | NULL | 用途 | 键 |  
| :----------- | :------ | ---- | ------ | ---- | -------- | ---- |  
| book\_id | bigint | 20 | 0 | 否 | 图书号 | ✔ |  
| name | varchar | 20 | 0 | 否 | 书名 | |  
| author | varchar | 15 | 0 | 否 | 作者 | |  
| publish | varchar | 20 | 0 | 否 | 出版社 | |  
| ISBN | varchar | 15 | 0 | 否 | 标准书号 | |  
| introduction | text | 0 | 0 | 是 | 简介 | |  
| language | varchar | 4 | 0 | 否 | 语言 | |  
| price | decimal | 10 | 2 | 否 | 价格 | |  
| pub\_date | date | 0 | 0 | 否 | 出版时间 | |  
| class\_id | int | 11 | 0 | 是 | 分类号 | |  
| number | int | 11 | 0 | 是 | 剩余数量 | |  
  
*#### 2. 数据库管理员表admin*| 名 | 类型 | 长度 | 小数点 | NULL | 用途 | 键 |  
| :------- | :------ | ---- | ------ | ---- | ------ | ---- |  
| admin\_id | bigint | 20 | 0 | 否 | 账号 | ✔ |  
| password | varchar | 15 | 0 | 否 | 密码 | |  
| username | varchar | 15 | 0 | 是 | 用户名 | |  
  
*#### 3. 图书分类表class\_info*| 名 | 类型 | 长度 | 小数点 | NULL | 用途 | 键 |  
| :--------- | :------ | ---- | ------ | ---- | ------ | ---- |  
| class\_id | int | 11 | 0 | 否 | 类别号 | ✔ |  
| class\_name | varchar | 15 | 0 | 否 | 类别名 | |  
  
*#### 4. 借阅信息表lend\_list*| 名 | 类型 | 长度 | 小数点 | NULL | 用途 | 键 |  
| :-------- | :----- | ---- | ------ | ---- | -------- | ---- |  
| ser\_num | bigint | 20 | 0 | 否 | 流水号 | ✔ |  
| book\_id | bigint | 20 | 0 | 否 | 图书号 | |  
| reader\_id | bigint | 20 | 0 | 否 | 读者证号 | |  
| lend\_date | date | 0 | 0 | 是 | 借出日期 | |  
| back\_date | date | 0 | 0 | 是 | 归还日期 | |  
  
*#### 5. 借阅卡信息表reader\_card*| 名 | 类型 | 长度 | 小数点 | NULL | 用途 | 键 |  
| :-------- | :------ | ---- | ------ | ---- | -------- | ---- |  
| reader\_id | bigint | 20 | 0 | 否 | 读者证号 | ✔ |  
| password | varchar | 15 | 0 | 否 | 密码 | |  
| username | varchar | 15 | 0 | 是 | 用户名 | |  
  
*#### 6. 读者信息表reader\_info*| 名 | 类型 | 长度 | 小数点 | NULL | 用途 | 键 |  
| :-------- | :------ | ---- | ------ | ---- | -------- | ---- |  
| reader\_id | bigint | 20 | 0 | 否 | 读者证号 | ✔ |  
| name | varchar | 10 | 0 | 否 | 姓名 | |  
| sex | varchar | 2 | 0 | 否 | 性别 | |  
| birth | date | 0 | 0 | 否 | 生日 | |  
| address | varchar | 50 | 0 | 否 | 地址 | |  
| phone | varchar | 15 | 0 | 否 | 电话 | |

*#### 7. 管理员信息表admin\_info*| 名 | 类型 | 长度 | 小数点 | NULL | 用途 | 键 |  
| :-------- | :------ | ---- | ------ | ---- | -------- | ---- |  
| admin\_id | bigint | 20 | 0 | 否 | 管理员证号 | ✔ |  
| name | varchar | 10 | 0 | 否 | 姓名 | |  
| sex | varchar | 2 | 0 | 否 | 性别 | |  
| birth | date | 0 | 0 | 否 | 生日 | |  
| address | varchar | 50 | 0 | 否 | 地址 | |  
| phone | varchar | 15 | 0 | 否 | 电话 | |  
*#### 8. 图书预约信息表appointlist*| 名 | 类型 | 长度 | 小数点 | NULL | 用途 | 键 |  
| :-------- | :----- | ---- | ------ | ---- | -------- | ---- |  
| ser\_num | bigint | 20 | 0 | 否 | 流水号 | ✔ |  
| book\_id | bigint | 20 | 0 | 否 | 图书号 | |  
| reader\_id | bigint | 20 | 0 | 否 | 读者证号 | |  
| appoint\_date | date | 0 | 0 | 是 | 预约日期 | |

5.2.2物理设计

我们使用Mysql Workbench将数据表导入到本地，如下所示：



### 5.2 功能设计

首先用户在为注册前需要进行注册，如果为管理员则选择管理员注册选项，如果为读者则选择读者注册选项，注册成功后即可登录进系统。

5.2.1管理员功能

5.2.1.1：管理员注册

@RequestMapping("/registerAdmin.html")

public String addAdminDo( Admin admin,RedirectAttributes redirectAttributes)

服务器接收到浏览器发送的registerAdmin.html请求，调用对应Service类的方法。如果成功创建管理员则将attributeValue赋为“管理员添加成功”，如果创建失败则将attributeValue赋为“管理员添加失败”，并且在页面上显示出来。

5.2.1.2：管理员查看书籍信息

@RequestMapping("/admin\_books.html")  
public ModelAndView adminBooks()

服务器接收到管理员在页面中发出的”/admin\_books.html”请求，调用bookService.getAllBooks()方法，从数据库中获取书籍信息。

5.2.1.3： 管理员修改书籍信息

@RequestMapping("/book\_edit\_do.html")  
public String bookEditDo(@RequestParam(value = "pubstr") String pubstr, Book book, RedirectAttributes redirectAttributes)

{

book.setPubdate(getDate(pubstr));  
if (bookService.editBook(book)) {  
 redirectAttributes.addFlashAttribute("succ", "图书修改成功！");  
} else {  
 redirectAttributes.addFlashAttribute("error", "图书修改失败！");  
}  
return "redirect:/admin\_books.html";

}

在BookController中接收到”/book\_edit\_do.html”请求以后，服务器将网页中传入的值传入数据库中进行相应的修改，并返回相应的结果信息。

5.2.1.3：管理员增加图书

@RequestMapping("/book\_add\_do.html")  
public String addBookDo(@RequestParam(value = "pubstr") String pubstr, Book book, RedirectAttributes redirectAttributes) {  
 book.setPubdate(getDate(pubstr));  
 if (bookService.addBook(book)) {  
 redirectAttributes.addFlashAttribute("succ", "图书添加成功！");  
 } else {  
 redirectAttributes.addFlashAttribute("succ", "图书添加失败！");  
 }  
 return "redirect:/admin\_books.html";  
}

BookController收到页面发出的增加图书的请求以后，将页面请求中附带的请求调用bookService.addBook方法进行传入数据库，并返回对应的结果。

5.2.1.4图书管理员删除图书

@RequestMapping("/deletebook.html")  
public String deleteBook(HttpServletRequest request, RedirectAttributes redirectAttributes) {  
 long bookId = Long.parseLong(request.getParameter("bookId"));  
 if (bookService.deleteBook(bookId)) {  
 redirectAttributes.addFlashAttribute("succ", "图书删除成功！");  
 } else {  
 redirectAttributes.addFlashAttribute("error", "图书删除失败！");  
 }

return "redirect:/admin\_books.html";  
}

LendController收到页面发出的删除图书的请求后，调用bookService.deleteBook方法对数据库中图书信息进行修改，并返回对应的结果。

5.2.1.5：图书管理员查看全部读者列表

@RequestMapping("allreaders.html")  
public ModelAndView allBooks() {  
 ArrayList<ReaderInfo> readers = readerInfoService.readerInfos();  
 ModelAndView modelAndView = new ModelAndView("admin\_readers");  
 modelAndView.addObject("readers", readers);  
 return modelAndView;  
}

ReaderController收到页面发出的”allreaders.html”请求以后，对应该方法调用readerInfoService.readerInfos()接口发出数据库请求，返回所有读者信息，并在页面上以表格的形式回显。

5.2.1.6：图书管理员查看某位读者个人信息

@RequestMapping("/reader\_info.html")

public ModelAndView toReaderInfo(HttpServletRequest request) {

ReaderCard readerCard = (ReaderCard) request.getSession().getAttribute("readercard");

ReaderInfo readerInfo = readerInfoService.getReaderInfo(readerCard.getReaderId());

ModelAndView modelAndView = new ModelAndView("reader\_info");

modelAndView.addObject("readerinfo", readerInfo);

return modelAndView;

}

ReaderController收到页面发出的”/reader\_info.html”请求以后，对应该方法调用readerInfoService.getReaderInfo()接口发出数据库请求，返回该读者个人信息，并在页面上以表格的形式回显。

5.2.1.7：图书管理员对读者信息进行删除

@RequestMapping("reader\_delete.html")  
public String readerDelete(HttpServletRequest request, RedirectAttributes redirectAttributes) {  
 long readerId = Long.parseLong(request.getParameter("readerId"));  
 if (readerInfoService.deleteReaderInfo(readerId) && readerCardService.deleteReaderCard(readerId)) {  
 redirectAttributes.addFlashAttribute("succ", "删除成功！");  
 } else {  
 redirectAttributes.addFlashAttribute("error", "删除失败！");  
 }  
 return "redirect:/allreaders.html";  
}

ReaderController收到页面发出的”/reader\_delete.html”请求以后，对应该方法调用readerInfoService.deleteReaderInfo(readerId) 接口发出数据库请求，根据页面中传递的参数进行数据库表项的修改。

5.2.1.8：图书管理员查看读者预约列表

@RequestMapping("/admin\_appoint.html")  
public ModelAndView Admin\_Appoint(){  
 ModelAndView modelAndView=new ModelAndView("admin\_appoint");  
 modelAndView.addObject("list",lendService.lendList());  
 return modelAndView;  
}

AppointController接受到页面发出的”/admin\_appoint.html”的请求以后，从appointlist数据表中获取所有表项，并且在页面上以表格的形式显示出来。

5.2.1.9：图书管理员同意读者预约请求

@RequestMapping("/appointbook.html")  
public String bookAppoint(HttpServletRequest request,RedirectAttributes redirectAttributes){  
 long bookId = Long.*parseLong*(request.getParameter("bookId"));  
 long readerId = ((ReaderCard) request.getSession().getAttribute("readercard")).getReaderId();  
 System.*out*.println(bookId);  
 System.*out*.println(readerId);  
 Appoint appoint = new Appoint();  
 appoint.setBookId(bookId);  
 appoint.setReaderId(readerId);  
 if(appointService.Appointbook(appoint)){  
 redirectAttributes.addFlashAttribute("succ","图书预约成功");  
 }  
 else{  
 redirectAttributes.addFlashAttribute("error","图书预约失败");  
 }  
 return "redirect:/reader\_book.html";  
}

图书管理员点击预约表中的某位读者后的同意按钮即可向后台发送”appointbook.html”请求，数据库将从预约表中删除该表项，并且在lend\_list表中加入读者信息和书籍信息。

5.2.1.10：图书管理员查看借阅表

@RequestMapping("/lendlist.html")  
public ModelAndView lendList(HttpServletRequest request) {  
 ModelAndView modelAndView = new ModelAndView("admin\_lend\_list");  
 modelAndView.addObject("list", lendService.lendList());  
 return modelAndView;  
}

LendController截取到页面发送给的“lendlist.html”请求，从后台数据库中读取lend\_list表项的所有的内容，并且在页面中以表格的形式显示出来。

5.2.1.11：图书管理员删除某读者的借阅记录

@RequestMapping("/deletelend.html")  
public String deleteLend(HttpServletRequest request, RedirectAttributes redirectAttributes) {  
 long serNum = Long.parseLong(request.getParameter("serNum"));  
 if (lendService.deleteLend(serNum) > 0) {  
 redirectAttributes.addFlashAttribute("succ", "记录删除成功！");  
 } else {  
 redirectAttributes.addFlashAttribute("error", "记录删除失败！");  
 }  
 return "redirect:/lendlist.html";  
}

LendController截取到页面发送的“/deletelend.html”请求，从后台数据库中删除lend\_list符合条件的表项，并且在页面中显示操作后的结果。

5.2.1.12：管理员修改个人密码

@RequestMapping("/admin\_repasswd\_do")  
public String reAdminPasswdDo(HttpServletRequest request, String oldPasswd, String newPasswd, String reNewPasswd, RedirectAttributes redirectAttributes) {  
 Admin admin = (Admin) request.getSession().getAttribute("admin");  
 long id = admin.getAdminId();  
 String password = loginService.getAdminPassword(id);  
 if (password.equals(oldPasswd)) {  
 if (loginService.adminRePassword(id, newPasswd)) {  
 redirectAttributes.addFlashAttribute("succ", "密码修改成功！");  
 return "redirect:/admin\_repasswd.html";  
 } else {  
 redirectAttributes.addFlashAttribute("error", "密码修改失败！");  
 return "redirect:/admin\_repasswd.html";  
 }  
 } else {  
 redirectAttributes.addFlashAttribute("error", "旧密码错误！");  
 return "redirect:/admin\_repasswd.html";  
 }  
}

LoginController截取到页面发送的“/admin/repasswd\_do.html”请求，修改Admin表中的数据项，并且在页面上返回操作结果。

5.2.2读者功能

5.2.2.1：读者查看图书列表

@RequestMapping("/reader\_books.html")  
public ModelAndView readerBooks(HttpServletRequest request) {  
 ArrayList<Book> books = bookService.getAllBooks();  
 ReaderCard readerCard = (ReaderCard) request.getSession().getAttribute("readercard");  
 ArrayList<Lend> myAllLendList = lendService.myLendList(readerCard.getReaderId());  
 ArrayList<Long> myLendList = new ArrayList<>();  
 for (Lend lend : myAllLendList) {  
 // 是否已归还  
 if (lend.getBackDate() == null) {  
 myLendList.add(lend.getBookId());  
 }  
 }  
 ModelAndView modelAndView = new ModelAndView("reader\_books");  
 modelAndView.addObject("books", books);  
 modelAndView.addObject("myLendList", myLendList);  
 return modelAndView;  
}

BookController截取到页面发送的“/reader\_books.html”请求，返回book表中所有额的数据项，并且在页面上以表格的形式返回结果。

5.2.2.2：读者查看图书详细信息

@RequestMapping("/reader\_book\_detail.html")  
public ModelAndView readerBookDetail(HttpServletRequest request) {  
 long bookId = Long.parseLong(request.getParameter("bookId"));  
 Book book = bookService.getBook(bookId);  
 ModelAndView modelAndView = new ModelAndView("reader\_book\_detail");  
 modelAndView.addObject("detail", book);  
 return modelAndView;  
}

BookController截取到页面发送的”/reader\_book\_detail.html”信息，然后在数据库中选取Book表中相应的表项，并将结果在页面中以列表的形式显示出来。

5.2.2.3：读者查询某书

@RequestMapping("/reader\_querybook\_do.html")  
public ModelAndView readerQueryBookDo(String searchWord) {  
 if (bookService.matchBook(searchWord)) {  
 ArrayList<Book> books = bookService.queryBook(searchWord);  
 ModelAndView modelAndView = new ModelAndView("reader\_books");  
 modelAndView.addObject("books", books);  
 return modelAndView;  
 } else {  
 return new ModelAndView("reader\_books", "error", "没有匹配的图书");  
 }  
}

BookController截取到页面发送的”/reader\_querybook\_do.html”请求，根据页面传输的关键字查询相应的图书信息，并且在页面上以条目的形式显示出来。

5.2.2.4：读者预约图书

@RequestMapping("/appointbook.html")  
public String bookAppoint(HttpServletRequest request,RedirectAttributes redirectAttributes){  
 long bookId = Long.*parseLong*(request.getParameter("bookId"));  
 long readerId = ((ReaderCard) request.getSession().getAttribute("readercard")).getReaderId();  
 System.*out*.println(bookId);  
 System.*out*.println(readerId);  
 Appoint appoint = new Appoint();  
 appoint.setBookId(bookId);  
 appoint.setReaderId(readerId);  
 if(appointService.Appointbook(appoint)){  
 redirectAttributes.addFlashAttribute("succ","图书预约成功");  
 }  
 else{  
 redirectAttributes.addFlashAttribute("error","图书预约失败");  
 }  
 return "redirect:/reader\_book.html";  
}

AppointController截取页面发送的”/appointbook.html”请求，对appointlist表进行插入操作，并且在页面中显示操作结果。

5.2.2.5：读者归还图书

@RequestMapping("/returnbook.html")  
public String bookReturn(HttpServletRequest request, RedirectAttributes redirectAttributes) {  
 long bookId = Long.*parseLong*(request.getParameter("bookId"));  
 long readerId = ((ReaderCard) request.getSession().getAttribute("readercard")).getReaderId();  
 if (lendService.returnBook(bookId, readerId)) {  
 redirectAttributes.addFlashAttribute("succ", "图书归还成功！");  
 } else {  
 redirectAttributes.addFlashAttribute("error", "图书归还失败！");  
 }  
 return "redirect:/reader\_books.html";  
}

LendController截取页面发出的“/return\_books.html”请求，并输出数据库中lend\_list表中符合条件的表项，在页面上显示操作结果。

5.2.2.6：读者查看借书日志

@RequestMapping("/mylend.html")

public ModelAndView myLend(HttpServletRequest request) {

ReaderCard readerCard = (ReaderCard) request.getSession().getAttribute("readercard");

ModelAndView modelAndView = new ModelAndView("reader\_lend\_list");

modelAndView.addObject("list", lendService.myLendList(readerCard.getReaderId()));

return modelAndView;

}

LendController截取到页面发送的”/mylend.html”请求，从数据库中的lend\_list表中选取符合条件的表项，并且以表格的形式回显到页面中。

5.2.2.7：读者修改密码

@RequestMapping("/reader\_repasswd.html")  
public ModelAndView reReaderPasswd() {  
 return new ModelAndView("reader\_repasswd");  
}  
  
@RequestMapping(value = "/reader\_repasswd\_do",method = RequestMethod.*POST*)  
public String reReaderPasswordDo(@RequestParam long readerId, @RequestParam String oldPassword, @RequestParam String newPassword,@RequestParam String reNewPassword, RedirectAttributes redirectAttributes) {  
 String password = loginService.getReaderPassword(readerId);  
 if (password.equals(oldPassword)) {  
 if (loginService.readerRePassword(readerId, newPassword)) {  
 redirectAttributes.addFlashAttribute("succ", "密码修改成功！");  
 return "redirect:/index.html";  
 } else {  
 redirectAttributes.addFlashAttribute("error", "密码修改失败！");  
 return "redirect:/reader\_repasswd.html";  
 }  
 } else {  
 redirectAttributes.addFlashAttribute("error", "旧密码错误！");  
 return "redirect:/reader\_repasswd.html";  
 }  
}

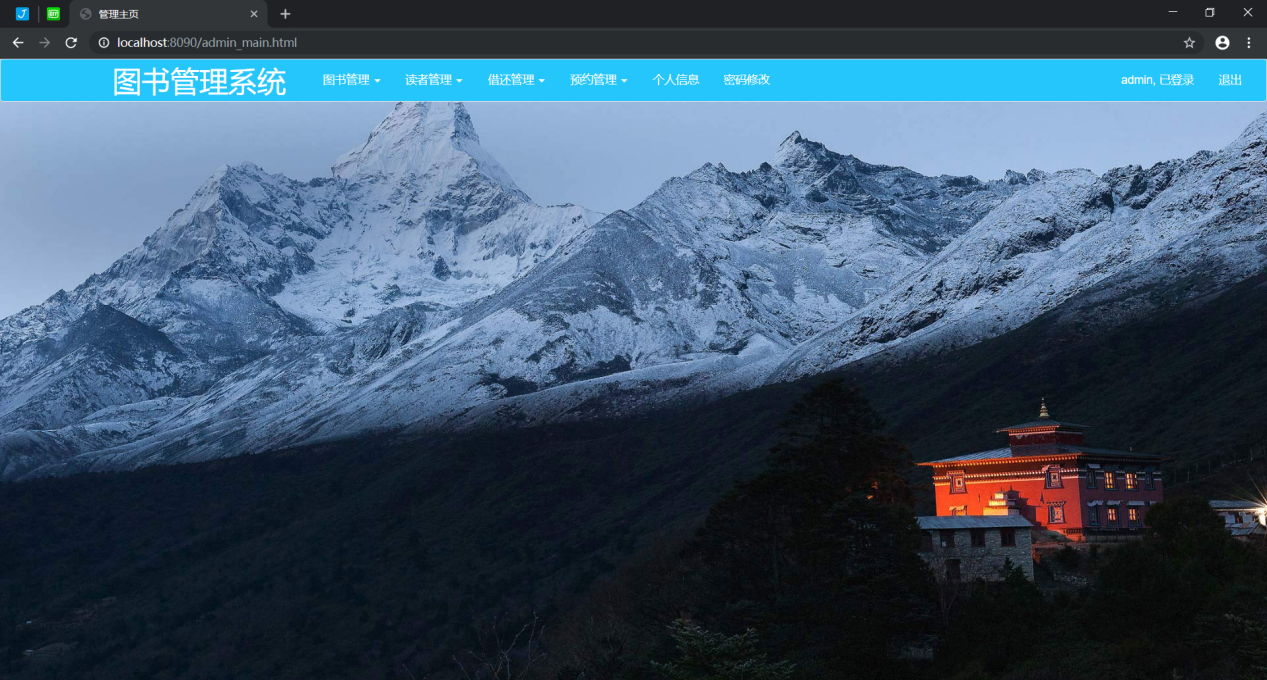
LoginController截取到页面发送的”/reader\_repasswd.html”请求，从数据库中的reader表中更改符合条件的表项，并且将操作结果回显到页面中。

5.3 UI设计

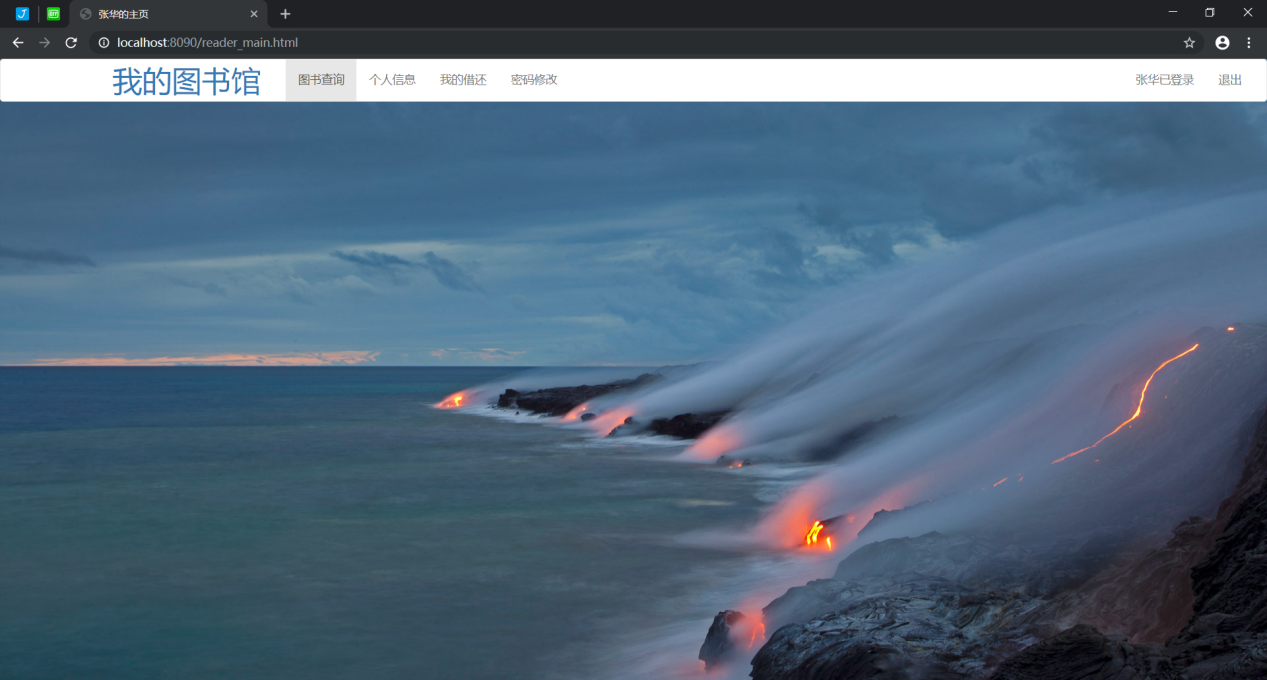
主页界面：



管理员界面:



读者界面：



# 六、实现的主要功能

七．总结

7.1亮点

这次数据库项目开发，我们使用了比较基础的SSM框架，从无到有，进行了大量的编码工作，实现了基本的数据库的增删查改等功能，并且在其基础上实现了读者对图书的预约及管理员对预约的同意这一较为复杂的交互动作。

在前端方面，为了页面的美观，我们特别地使用了Bing上提供的大量的高清壁纸，使得整个页面看起来令人美观大方，同时带有优美的意境。另外对于大量的信息显示我们均使用表格进行格式控制，使得信息看起来井然有序，同时也一目了然。

在数据库方面，为了实现高效的建表，我们使用了Mysql自带的图形化界面——mysql workbench，进行数据表的建立以及数据的导入。在数据导入时，利用了.csv文件可以直接整体导入数据库的特点，直接在excel进行数据的规整化，之后直接导入到数据库中。其中数据的爬取我们也使用了软件进行辅助，直接从电商平台中售卖的书籍中爬取相关的信息，由于python程序强大的功能，可以直接将数据保存为.excel文件的格式，实现了海量数据的自动化操作，无须使用人工进行操作。

在前后端的交互方面，代码的编写的思路大部分都是相通的，所以触类旁通，一旦能够实现一个功能，其余的功能也都类似。最大的亮点可能就在于预约功能的实现了，因为预约功能的整体流程为：读者发出预约请求，管理员同意预约请求。在后台数据库则表现为预约表的增加、预约表的删除和借阅表的增加。由于是在系统完成以后才发现读者不可以直接借书，而只能预约图书，所以我们使用了软件工程的原型开发模型，逐渐增加功能，每完成一项功能即进行测试与评价，所以开发效率较高。

7.2需要完善之处

首先是在演示的时候老师给予我们的建议都十分的中肯。

第一个不足之处在于我们选取的SSM框架的封装度不高，特别是前端代码依旧是使用.jsp文件进行书写，导致了代码冗长且显示效果并不美观，所以我们的前端人员花了很大功夫做出来的页面依旧不尽如人意。

第二个问题在于没能实现分页显示或者是动态分页效果。原因在于后端开发使用的java代码的逻辑也比较复杂，大量的接口导致编码的思路的混乱，再加上由于分页或者是动态显示的实现需要使用到大量的新的接口进行实现，所以在短短一周时间内没能将预约功能和动态页面显示功能完全完成。

第三个问题在于我们在开发时未能使用类似于github这种代码管理工具，所以导致项目配合度不高，代码无法及时性的进行交互和调试。这也是我们开发效率比较低的原因之一。

这次项目虽然结束了，但是它带给我们的经验是宝贵的。上面提出的不足之处都是在以后做项目开发的时候需要避免的。不得不说，方法比努力更加重要，当然两者是缺一不可。但是如果没有找对方法，做事就会事倍功半，尤其是软件开发方面，更需要精简开发过程，才能实现人力物力损耗最小化的目的。