Author

xx@gmail.com

Library Management System

Abstract

The system differs greatly from the traditional library management system, it covers all functions of the traditional system, and it appeals to students and teachers because of its new concept, living library, also known as human library, also, it is very useful due to its multiple features such as micro-message reminder, used book trade, recommendation, etc.  
You can get a detailed critical review of the project from this document.

Table of Contents

[1 Objectives 3](#_Toc399012315)

[2 Scope and Constraints 4](#_Toc399012316)

[3 Project Details 5](#_Toc399012317)

[3.1 Background 5](#_Toc399012321)

[3.2 Problem Statement 6](#_Toc399012322)

[3.3 Solution 7](#_Toc399012323)

[4 User Requirements 9](#_Toc399012324)

[4.1 Use Cases 9](#_Toc399012325)

[5 Architecture 17](#_Toc399012326)

[6 Detailed Design 21](#_Toc399012327)

[6.1 Class Design 21](#_Toc399012328)

[6.1.1 Data Layer Design 21](#_Toc399012329)

[6.1.2 Logic Layer Design 23](#_Toc399012330)

[6.1.3 Service Layer Design 24](#_Toc399012331)

[6.1.4 Presentation Layer Design 25](#_Toc399012332)

[6.2 Database Design 26](#_Toc399012333)

[6.2.1 Book Table 26](#_Toc399012334)

[6.2.2 Category Table 27](#_Toc399012335)

[6.2.3 User Table 27](#_Toc399012336)

[6.2.4 Dictionary Table 28](#_Toc399012337)

[6.2.5 Borrow Book Table 28](#_Toc399012338)

[7 Development Process 30](#_Toc399012339)

[7.1 Version 1.0 – Groundwork 30](#_Toc399012340)

[7.2 Version 1.1 – Basic Borrow 40](#_Toc399012341)

[7.2.1 Search a book 40](#_Toc399012342)

[7.2.2 User information 41](#_Toc399012343)

[7.2.3 Search a living book 42](#_Toc399012344)

[7.2.4 Borrow, return and renew 43](#_Toc399012345)

[7.2.5 Used book trade 46](#_Toc399012346)

[7.3 Version 1.2 – Ranking and Recommendation 47](#_Toc399012347)

[7.4 Version 1.3 – Notification 52](#_Toc399012348)

[7.5 Version 2.0 – Login with QR code 59](#_Toc399012349)

[7.6 Version 2.1 – Publish to web 68](#_Toc399012350)

[8 Testing 72](#_Toc399012351)

[8.1 Functional Test 72](#_Toc399012352)

[Test Cases 72](#_Toc399012353)

[8.2 Usability Test 78](#_Toc399012354)

[9 Conclusion 84](#_Toc399012355)

[9.1 Technical Complexity 84](#_Toc399012356)

[9.2 Innovation 84](#_Toc399012357)

[9.3 Future Enhancements 85](#_Toc399012358)

[10 Appendices 87](#_Toc399012359)

[10.1 Reference Guide 87](#_Toc399012360)

[10.1.1 Books 87](#_Toc399012361)

[10.1.2 Websites 87](#_Toc399012362)

[10.1.3 Our online library management system address 87](#_Toc399012363)

[10**.**2 Usability test questionnaire 88](#_Toc399012364)

[10.3 Application form of living book 92](#_Toc399012365)

# Objectives

The Library Management System aims to provide an online library management for the students and teachers of Jiangsu Second Normal University to search, borrow, return, comment and rate books, also, it aims to build an online community for them to share knowledge, recommend books, as well as trade used books.

The system introduces a new concept called Living Library, also known as Human Library, which started in Denmark that takes human as an education tool or knowledge carrier for people to borrow, you may talk with these borrowed people about their specific areas, skills or experience so that you can get firsthand materials of the theme you want to explore, or fresh impression about the things you’ve learned before, or brand new understanding about a certain kind of people, etc.

The system is designed to make the living library easier to apply, it accepts reservation and allows 2 types of talking with the borrowed people, face-to-face talk and online talk using Skype. It also supports searching, commenting and rating just as the books.

In addition, the system will make it more efficient and convenient for administrators to monitor books, by looking up the rate, comments and borrowed count of the book, they can track users’ needs and take some action to meet the needs. For example, they may collect more demanded books if a book is highly recommended but there are few to lend, or they will consider to buy more books that are the same kind as the hot books.

What’s more, the system will focus on user experience and will promote the usage of the library, for instance, the system will develop a module of notification that combines e-mail, micro-blog, and wechat (like twitter and MSN) for better user experience and higher usage of the library. It will bring a revolution to the library of Jiangsu Second Normal University.

# Scope and Constraints

The Library Management System provides services of borrowing, renewing, returning, commenting and rating books. It allows students and teachers to borrow living library via its online reservation system. It also builds a platform for trading used books.

The system is not redesigning the old library management system, it will select a few of books that are valuable and useful to suggest a high-quality reading for users.

For better user experience and higher usage of the library, the system will have its own notification module, it will combine e-mail, micro-blog and wechat together to make a good reminder for user to manage books.

Due to budget constraint as well as the space constraint of the system, it will not provide user’s personal space in this version, but just a simple page for sample, and this could be a future enhancement of the system.

# 3 Project Details



## Background

Project Background:

Located in Nanjing, Jiangsu province, Jiangsu Second Normal University is a full-time undergraduate school. At present, it has 16 institutes and nearly 7,000 full-time students. In recent years, the university is developing rapidly with enlarging dimensions. However, the traditional library management system is out of date, it’s running slowly, and it’s offline that only provides service inside the library which is a great limit for students who expect to inquire books in and out of the library at any time, as a matter of fact, the old library management system leads to very low usage of the library. Compared with many online systems, the old system lacks interaction with users, and the functions seem to be too simple to meet the various needs of users.

After talking with our sponsor about our observation and ideas, we finally get a support to develop a new library management system for the university.

The system is a not redeveloped library management system, but a new system to attract more students to participate and learn from each other by sharing reading experience and giving guides to green hands. It also brings convenience to students to store personal data (as a future enhancement) and trade their used books. Moreover, they can book human books and get precious inspiration or tutorials because of online living library reservation. On the other hand, compared with the old fashion way, librarians, now play the role of system administrators, are able to interact with users, and manage books more positively and efficiently according to books’ comments and rates. With increasingly participation of students on the new system, more and more students will find their interests in reading and studying. We have the faith that one day, the new system will totally replace the traditional library management system.

Sponsor Background:

Located in Nanjing, Jiangsu Province, Jiangsu Second Normal University is a full-time undergraduate school. Covering an area of about 35 hectare, it currently has three campuses of CaoChangMen, PuKou and XiaoYing. Besides normal education, it also opens some majors for Jiangsu economic and social development. At present, it has 16 institutes and nearly 7,000 full-time students. The school has three libraries, collecting about half million books covering almost all subjects. The libraries grow from two locations within 20 staffs to its current three locations within 32 staffs. Each library provides private learning rooms and public learning areas.

Living Library Background:

Living Library, also known as Human Library, is started in Denmark. The core idea is taking human as an education tool or knowledge carrier to enhance people’s participation. These borrowed people may be male and female, old and young, and many of them have controversial social statuses like transgendered people, radicals, AIDS patients, homosexuals, Mohammedans, strippers and alcoholics. People may borrow these living books and talk with them, thus they can learn each other better and reduce the prejudice and discrimination in the society.

## Problem Statement

Although the library management system is still available in Jiangsu Second Normal University, it is out of date. A lot of books are in low rate of usage because of the isolation of system and lack of updated statistics. Users can only browse books inside the library, that is, the old system is offline. There are a lot latest books’ information also need to be updated into the system. The database of the system doesn’t have backup service and the processing speed is low when a lot inquires executed simultaneously.

From the survey (see appendix) which we aim to find out needs for future library management system and drawbacks for the current one, we know that the majority of students think they should be able to search books outside the library and keep up with the information of latest books. They want to share reading experience and rate the books they’ve read just like what they can do at Amazon book market. They also want to have their own space to store personal electric documents.

On the other hand, they complain that the current system is boring, too simple to meet the needs, and the UI design is bad, and there is no way to find out the latest or most popular books, that the library doesn’t provide activities or any kind of communication between readers. They expect to get some tutorial or some kind of guide as well.

Based on all the facts of current system and the result of the survey, we decide to develop an online library management system for the JSNU (Jiangsu Second Normal University) which is more suitable for current situation.

Before we start, we ask ourselves some critical questions as the table shows below:

|  |  |
| --- | --- |
| 1 | How to design a reasonable system architecture to make all subsystems work? |
| 2 | How to apply the concept “living library” to a real online platform? |
| 3 | How to design an attractive and user-friendly UI? |
| 4 | How to protect users’ privacy (e.g. Skype ID is only visible when a reservation is completed)? |
| 5 | How to make use of some external APIs (e.g. Java Email service or even SMS service)? |
| 6 | How to build up a reminder to notify users when books are about to overdue, or a reservation has been made or cancelled? |
| 7 | How to design used books trading platform with some suitable restrictions or rules to prevent spamming? |
| 8 | How to sort the books based on rates, popularity, or the number of comments? |
| 9 | How to use some web security strategies to protect the safety of the system? |

Figure 3.1 Problem Statement

## Solution

Living library is a total new concept for students and teachers in Jiangsu Second Normal University, my sponsors show great interest in this idea and is willing to provide me any useful resources for designing the library (e.g. Study rooms for communication, some candidate teachers may become living books).

As far as I am concerned, there are only two organizations providing service of living library, one is Douglas College, the other is Coquitlam Library. The form of these living libraries is simple. They offer some activities, inviting some typical and critical people (e.g. ex-offenders, HIV carriers and gays) and sort them into different titles of books for people to borrow. After a talk with the living book, readers can learn from their stories or experience, reduce prejudice and remove stereotype.

However, the current living library has some limits :

Limits

Figure 2 Limits of Living Library

Firstly, there are very few of book categories, most of them are centralized in very critical and typical area. On the other hand, living books related to academic research areas are almost zero. Thus the mode of the traditional living library is not suitable for JSNU. It can hardly make contribution to the students’ academic study.

Secondly, the reservation of traditional living library only stays on massive paper work, it’s isolated and has limited communication form among library administrators, users and living books. It can hardly attract students to participate.

Thirdly, the traditional living library lacks availability. Users can only borrow living books during library’s activities. If there is no activity, there is no chance to borrow any living books. It is inconvenient for students to study constantly.

Fourthly, it is hard for librarians (system administrators) to track performance of each living book due to lack of users’ feedbacks.

Last but not least, the form of using living library is single, it only allows for talking face to face, there are no supports for people chatting via online chatting tools such as Skype or MSN. When the users or the living books encounter time or place problems, chatting online will be a good way to solve the problem.

Above all, the limits show that the current living library is not enough for JSNU, so we think about a solution to apply the living library to our system. The detailed solution gives out at chapter 9.

# 4 User Requirements

## 4.1 Use Cases

First, we concluded a list of user requirements according to our survey, observation at the library, and talking with some librarians as well, here is what we concluded:

The system is required by these stick-holders: users, administrators, and living books

Users’ Requirements:

1. Users can register with valid student ID through the website.
2. Users can edit personal information.
3. Users can search a book according to its name, category, publish time, press, author, book ID, etc.
4. Users can sort books according to borrowing times, the number of comments or rates.
5. Users can borrow a book via the system; they will received a confirmation letter when they successfully complete a booking, then, they can pick up a book with a valid student ID.
6. Users can renew the borrowed books, however, renewing is only allowed once and duration is a month.
7. Users can rate and comment books when they successfully return books.
8. Users can post sale information of used books by simply clicking “I want to sell used books”.
9. Users can post demand information of used books by simply clicking “I want to buy used books”.
10. Users can search used books.
11. Users can upload pictures of used books.
12. Users can edit or delete their posted information.
13. Users can contact a buyer or seller by clicking “contact” button and they can communicate by emails. For protection of privacy, real email addresses of buyers and sellers should be hidden by using anonymous email addresses.
14. Users can search human books according to different subjects (e.g. math, computer, history, and physics).
15. Users can sort human books according to borrowing times, the number of comments or rates.
16. Users can borrow a human book via the system and choose its available schedule; two forms of meeting is optional for users: online meeting (via Skype) or offline meeting.
17. Users submit meeting form and meeting theme before booking living book.
18. Users can contact wanted human books by simply clicking “contact” button. They can communicate by email. For protection of privacy, real email addresses of buyers and sellers should be hidden by using anonymous email addresses.
19. Users can cancel a booking.
20. Users can rate and comment human books when they finished reading.

Administrators’ Requirements:

1. Administrators can add, delete and edit books
2. Administrators can delete over-offensive comments.
3. Administrators can confirm a booking when users pick up with a valid student ID, however, a booking is only hold up 48 hours before its cancelation.
4. Administrators can confirm a returning of books when they receive them.
5. Administrators can delete posts irrelevant to the used books.
6. Administrators can add human books, delete human books or edit information of human books.
7. Administrators can add users to blacklist based on following situations:
   1. Users don’t return books or don’t return books on time.
   2. Users often post junk information irrelevant to used books.
   3. Users often fail to keep the appointment with reserved human books.
   4. Users often cancel the meeting with human books.
   5. Users often post malicious comments.

Living Books’ Requirements:

1. Living books can edit personal information. (Such as the title, subject, special filed, interests, introduction, available schedule for booking, reading forms, contact information and etc.).
2. Living books can cancel the reservation made by users.
3. Living books can comment the users.
4. Living books can send and receive the letter inside station.

With the conclusion list of the requirements, we analyze them using UML and try to make it clearer for the system to be understood, in this phrase, we find that the notification of system should be put on the table to meet the needs. In UML, such a **notification of system** is also treated as a user. Here are what we analyzed in UML:



Figure 3 Users' Use Case

As it shows in the figure, users mainly have 4 use cases, borrow books, borrow living books, trade used books, and manage personal information, these 4 cases have extended cases, or son cases, and some of them share one or more extended cases because of the same logic. Then we analyze each case, mainly the stimulation and respond sequence of the use case.

**Stimulation/Respond Sequence:**

1. User searches a book by the category, title, publishing time, author, ISBN etc.
   1. The system shows the result of searching in a list
2. User sorts books according to the reading count, rating score, and the adding time.
   1. The system gives out the sorting result
3. User borrows a book online
   1. The system checks if the user could borrow
   2. If so, sends a success message to the user
   3. If not, sends a failure notification to the user.
4. User renews a borrowed book
   1. The system checks if the user could renew the book
   2. If so, sends a success message to the user
   3. If not, sends a failure notification.
5. User collects a book
   1. The system checks if the book is already in the collection of the user
   2. If so, sends a notification
   3. If not, add the book to its collection list.
6. User comments a book
   1. The system checks if the user could comment the book (the user is not in the blacklist)
   2. If so, add the comments to the book
   3. If not, gives out a notification to the user.
7. User reserves the living book
   1. The system checks if the reservation is valid (all required information are filled and valid, and the user is allowed to make a reservation, that is, the user is not in the blacklist)
   2. If so, sends a success message to the user and the living book ( the message includes the meeting time, meeting form, the theme to be discussed, Skype ID if meeting form is online, room number if meeting form is offline )
   3. If not, sends a notification to the user.
8. User cancels the reservation
   1. The system checks if it is 6 hours ahead of the reserved time
   2. If so, cancel the reservation, sends a success message to the user and the living book
   3. If not, gives a notification to the user that he may not cancel it.
9. User posts used book for sale
   1. The system checks if the information is valid and completed (title, ISBN, author, quality, price, etc.)
   2. If so, post it to the page of used book trading
   3. If not, gives hints to the user
10. User posts used book for purchase
    1. The system checks if the information is valid and completed (title and author)
    2. If so, post it to the page of used book trading
    3. If not, gives hints to the user
11. User communicates with the other user on the used book trading platform
    1. The system pops a dialog to require the sending content

11.1 User fill in the blank and click send

* 1. The system will send a letter inside station and an e-mail as well to the other user and his default mailbox

1. User registers
   1. The system shows a page to require the basic information of the user ( name, birthday, gender, student number or faculty number, hobby, email address, Skype ID, password and the questions of finding password)

12.1 User fills in the blank

* 1. The system checks if the information is valid and completed
  2. If so, the system sends out a success message to the user
  3. If not, sends a notification to the user

1. User logs in
   1. The system checks if the name and password are correct
   2. If so, jumps to the page with user information
   3. If not, gives hints to user for incorrect name or password
2. User logs out
   1. The system save the data of the user and jumps to the default page without user information
3. User modify personal information
   1. The system checks the information and saves the modified information



Figure 4 Administrators' Use Cases

So administrators mainly do the management work, they manage books, living books, posts, comments and users, often the management work deals with adding, deleting, modifying data, so is the work in our system, and there’s no need to illustrate any more, what we focus on is some special operations admins can do in our system, we analyze these operations with stimulation and respond sequence shown as below.

**Stimulation/Respond Sequence:**

1. Admin confirms a borrowing after checking the users’ status
   1. The system checks the book’s state then shows a message of success or notification
   2. If the book is borrowed, and the user doesn’t come to fetch, then the system will cancel the borrowing, that is, without a confirmation of admin within **2 days**, the borrowing will be canceled automatically by system
2. Admin confirms a returning after checking the book
   1. The system will modify the book’s state and shows a message
3. Admin adds users to blacklist
   1. The system shows a list of blacklist with add and remove buttons

3.1 Admin clicks add button

* 1. The system pops a dialog to require the user’s ID

3.2 Admin enters the user id

* 1. The system searches the user id, shows the information of the user

3.3 Admin clicks confirm button

* 1. The system adds the user to its blacklist, and the user is now shown on the list



Figure 5 Living Books' Use Cases

The living books use the system in a simple and straight way, they only need do the interaction with the user and manage their personal information especially their time tables which are very important for the whole process of borrowing living books. The modify information case has one special feature, that’s the time table, how to design the time table to make it easy for living books to modify and for users to select is critical, but on this phrase, we don’t need to care about that. The comment user case is just like what it does in the Users’ Use Case, so we don’t analyze repeatedly. The cancel reservation case is the same as what a user does in canceling a reservation except the time limit, the stimulation and respond sequence is:

Living book cancels the reservation,

1. The system checks if it is 2 hours ahead of the reserved time
2. If so, cancel the reservation, sends a success message to the user and the living book
3. If not, gives a notification to the living book that he may not cancel it.

As for the send and receive letter inside station case, actually it is a feature of the system, that is, all the users, admins and living books can use the letter inside station, but we put it here to press the duty to interact with users of living books.



Figure 6 Notification Use Cases

This is what we concluded from the stick-holders’ statements and the systems’ features, for better user experience, we put the notification system on table, we hope that we can notify users in time, so the letter inside station and email are not enough, we think about combining the wechat and the system, so that when there are books to be overdue or a reservation cancelled, the users and the living books can get a timely message with the notifications.

# 5 Architecture

The system makes use of the Java lightweight open source frameworks Struts2, Spring and Hibernate, MySql for database management, Tomcat for web server, and MyEclipse for developing. The architecture of the system is shown as the following figure:

Hibernate support

Struts2 support

Service Layer

Data Layer

Presentation Layer

Figure 7 System Architecture

As shown in the figure, the service layer of the system uses Spring, the data layer uses Hibernate that can be integrated into Spring via supporting classes of Spring. The presentation layer uses Struts2 to pass data and control the web pages written in JSP, EL, and JSTL. The whole system uses the 3 layer architecture, which allows only the upper layer calls the lower layer, so as to achieve loose coupling among layers.

Struts2 framework:

Struts2 is a Web application framework based on MVC. The core meaning of MVC pattern is MVC pattern decoupling, dividing the entire application into three parts, model, view, and controller. It tightly controls the communication between the three parts, in order to obtain a clear-structured, function-distribution-reasonable, reusable, extensible, and maintainable applications. By using Struts2 to control the page jumping, you don’t have to write complex code written in Servlet, the robust value stack and OGNL expressions of Struts2 can be used for transmission and control of data.

For enterprise applications, server-side validation is necessary, strict check should be taken on the user provided data before any business logic codes are called, in tradition, data validation needs programmers writing code to achieve, and often the codes are mixed with business logic codes. However, Struts2 provides a framework for data checking, and you can easily tell which codes are for data validation and which are for business logic because it separates the data validation and the business logic. Also, Struts2 has powerful tag library and filters, all of the factors improve the efficiency for the enterprise application development.

The operation process of Struts2 is very simple, when the user request arrives at FilterDispatcher controller, the controller will execute the corresponding Action according to the submitted URL and configuration in the struts.xml. Struts2 Action realizes the decoupling with Servlet API, Action does not require any class inheritance or interface implementation. After finishing processing the user request, Action will jump to the page that is pre-configured in Result according the processed result, and display the data caught in Action to user.

Hibernate framework:

Hibernate is a persistent software based on Java open source, it encapsulates a lightweight package to provide JDBC, ORM (Object Relational Mapping) service. ORM automatically maps the objects of the program to tables of a relational database according the metadata which describes the mapping between objects and tables of database. If we use JDBC to connect to the database and do some operation on database, we need to write a lot of code, while using Hibernate we only need to configure the mapping between Java entity class and the relational database tables, and the method for Hibernate to query and acquire data, thus reduce the amount of code, and improve the efficiency of development.

Hibernate provides one to many, many to one, many to many relationships of objects association, as well as immediate loading and delayed loading support, which facilitates the manipulation of data. The object-oriented HQL query will generate the corresponding SQL statement according to the mapping relationship between objects and database tables, which simplifies the complex SQL statements writing. Hibernate also provides a 1 level cache and 2 level cache, the rational use of 2 level cache can reduce the number of database access effectively which will enhance the overall performance of the system.

Spring framework:

Spring is a lightweight open-source framework of Java SE/EE application, with IoC (Inverse of Control) and AOP (Aspect Oriented Programming) as the core. Inverse of control is also called dependency injection, it makes the object class a passive receiving dependent class rather than class that need to find service on their own. Dependency injection gives the control of dependencies among objects to Spring, so you don’t need to worry about when to instantiate an object, and just focus on the business logic. Spring also provides a powerful support to the Aspect Oriented Programming, by separating out the business logic from application services, it achieves cohesion development.

To integrate Spring and Hibernate, we can put the SessionFactory interface of Hibernate that manage the data access to the IoC container of Spring, so that we only need configure the file rather than manually create an instance of SessionFactory when Hibernate accesses the database. We can also use the transaction mechanism of Spring, so as to switch different data sources without modifying the source code.

To integrate Spring and Struts2, the instantiating the Struts2 Action no longer needs to be managed by Struts2, but managed by the IoC container of Spring instead, thus we can reduce the coupling procedures, and separate the controller and the business logic, which brings great convenience to future maintenance and expansion.

ExtJs:

ExtJs is an independent of back-end, JavaScript written Ajax framework. The power of ExtJs lies in its various components, you can build rich and colorful front pages use the components. ExtJs form controllers are perfect, they support functions as sorting, caching, draging, hiding and editing data, etc. Form controllers support not only the beautiful appearance but also support data check at the front end. Layout controllers can take charge of the entire page layout without writing a lot of code in JavaScript. In addition, ExtJs can reduce as much as possible of the count of jump pages, or even eliminate the need for page jumping.

MyEclipse:

MyEclipse is a very good J2EE tool for integrated development based on Eclipse, it has powerful collection of plugins supports than Eclipse, and better support the open source products. It is an enterprise level development platform, an extension of Eclipse. In MyEclipse, database and J2EE development can be integrated easily, as for the program compiling, running, testing, deploying and releasing, they can all be finished in MyEclipse. Besides, MyEclipse has a good support for the Java open-source framework like Struts2, Spring and Hibernate. Another commonly used integrated development tool is called Intellij IDE, but it lacks plugins, and occupies large memory. Therefore, we choose MyEclipse as the top tool for J2EE development to speed up the development of J2EE applications.

Optimizations for the system:

* **Apache Tomcat web server and load balancer**

The library management system should allow large number of queries processing parallel, therefore single server cannot satisfy the need of loading, what’s more, if the server fails to do normal operation, the whole system ducks. So we must find out the way to ensure that the system doesn’t crush easily. We choose to use the Tomcat cluster technology to solve the problems.

The Tomcat cluster can be achieved by two or more server software instances running on one or more server computers, the servers together make it transparent to clients, the clients see only a high available service. The reverse proxy cluster system use Apache to achieve load balancing, Apache will do the dispatching work, it receives the client’s request and then forwards the requests to different Tomcat servers, thus make the loading balanced. You can clearly see the working principle of the Tomcat cluster.

集群系统由一台或多台服务器计算机上运行的两个或更多服务器软件实例组成，这些服务器计算机彼此协同合作以透明地服务客户端的请求，从而从客户端角度看，整个集群租是一个高可用性服务。网站的集群系统使用Apache的反向代理来实现负载均衡，Apache会将用户的请求分别转发给不同的Tomcat服务器，以此来实现Tomcat的集群。集群部署图如图所示。

* **Cache of Hibernate**

The cache is located between applications and physical data source at the computer memory, its purpose is to reduce the number of applications accessing to physical data sources, so as to improve the performance of applications. When a program needs to query the database, it will first look up the data in cache, if hit, that is, if find the data, then it will not have to access the database anymore.

* **Partition and creating index for database**

A partition is a division of a logical [database](http://en.wikipedia.org/wiki/Database) or its constituent elements into distinct independent parts. Database partitioning is normally done for manageability, [performance](http://en.wikipedia.org/wiki/Optimization_(computer_science)) or [availability](http://en.wikipedia.org/wiki/Availability) reasons.

The partitioning can be done by either building separate smaller databases (each with its own [tables](http://en.wikipedia.org/wiki/Table_(database)), [indices](http://en.wikipedia.org/wiki/Index_(database)), and [transaction](http://en.wikipedia.org/wiki/Database_transaction) [logs](http://en.wikipedia.org/wiki/Database_log)), or by splitting selected elements, for example just one table. Horizontal partitioning (also see [*shard*](http://en.wikipedia.org/wiki/Shard_(database_architecture))) involves putting different rows into different tables. Perhaps customers with [ZIP codes](http://en.wikipedia.org/wiki/ZIP_code) less than 50000 are stored in CustomersEast, while customers with ZIP codes greater than or equal to 50000 are stored in CustomersWest. The two partition tables are then CustomersEast and CustomersWest, while a [view](http://en.wikipedia.org/wiki/View_(database)) with a union might be created over both of them to provide a complete view of all customers.

Vertical partitioning involves creating tables with fewer columns and using additional tables to store the remaining columns. [Normalization](http://en.wikipedia.org/wiki/Database_normalization) also involves this splitting of columns across tables, but vertical partitioning goes beyond that and partitions columns even when already normalized. Different physical storage might be used to realize vertical partitioning as well; storing infrequently used or very wide columns on a different device, for example, is a method of vertical partitioning. Done explicitly or implicitly, this type of partitioning is called "row splitting" (the row is split by its columns). A common form of vertical partitioning is to split dynamic data (slow to find) from static data (fast to find) in a table where the dynamic data is not used as often as the static. Creating a view across the two newly created tables restores the original table with a performance penalty, however performance will increase when accessing the static data e.g. for statistical analysis.

Creating index for database can improve the speed of data retrieval operations on a [database table](http://en.wikipedia.org/wiki/Table_(database)) at the cost of additional writes and storage space to maintain the index data structure. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more [columns of a database table](http://en.wikipedia.org/wiki/Column_(database)), providing the basis for both rapid random [lookups](http://en.wikipedia.org/wiki/Lookup) and efficient access of ordered records.

# 6 Detailed Design

## 6.1 Class Design

With the analysis of the user requirements and architecture as well, we now move on to the phrase of detailed designing, the system has 4 layers, namely are presentation layer, service layer, logic layer, and data layer, and we build these 4 layers with SSH (Struts2, Spring and Hibernate) framework.

### 6.1.1 Data Layer Design

The data layer includes models and DAO. Models are the data structure, or the single objects for the system to pass and process. Let’s take the model of book for example, its private properties like author, press, title, ISBN are the corresponding keys of the book table in the database, and its methods are getters and setters for these data properties like getAuthor, setAuthor, getPress, setPress, getTitle, setTitle, getISBN and setISBN, etc.

We use class diagram to illustrate the model design. Here’s the figure:



Figure 8 class diagram for model

DAO is for data access object, which is responsible for accessing data of the database, in the system, we make use of Hibernate framework to support DAO.



Figure 9 Class Diagram For DAO

### 6.1.2 Logic Layer Design

Logic Layer is for the business logic, it’s the core of the system, and it uses the resources of the system to make the system work like the real world. In the Library Management System, we name the layer Action out of habit. Action does the logical operations on the data by the support of Spring and Struts2, here’s what we designed:



Figure 10 Class Diagram for Logic Layer

### 6.1.3 Service Layer Design

Service layer is responsible for the interaction between logic layer and data layer, supported by Spring. It decouple the layers and make the system clear and maintainable. Here’s what we designed on the service layer:



Figure 11 Class Diagram for Service Layer

### 6.1.4 Presentation Layer Design

Finally, the presentation layer deals with UI and calls the logic layer, we use Struts2 for support. Since we use Dynamic Web Project in MyEclipse to develop the system, we need to add tags to the *web.xml* file under the WEB-INF folder:

|  |
| --- |
| <filter>  <filter-name>struts2</filter-name>  <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |

Figure 12 Add tags to support Struts2

## **6.2 Database Design**

Database is the cornerstone of the entire system, database design directly affects the merits of the success or failure of the entire system design, database design is described in this section.   
Database design is to convert solid models and needs of the real world into the process model database, which is the establishment of a database application system core issues. Database and application performance are based upon good database design, database data is the foundation of all operations, if the database design is not good, then all the other methods used to improve database performance is limited effective. The key database design is how to make the design of the database can reasonably store user data that is easy for user data processing.   
Database design must follow certain rules, in a relational database, this rule is the paradigm, and the paradigm is in line with the set of a certain level of relationship patterns. It follows the general design of the database third paradigm. Namely: database table does not contain non-primary key information that is is included in the other tables. Adopted paradigm reduces data redundancy, saving storage space, while speeding up the adding, deleting, and modifying data.

### 6.2.1 Book Table

Book table is mainly used for storing library books in the possession of the relevant information, including relevant information at the time of book storage adding by the administrator, this table is mainly used for readers and librarians to check books in the library system according to user- queried property of books, readers and librarians will be aware of other relevant information, like ISBN, which is to help readers to find books on the shelves to borrow, the book price is used for the compensation basis when the reader inadvertently lost books. Concrete table shown in Table 6.1.

Table 6.1 Book Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Whether allowed to be empty | Description |
| ID | text | 20 | No | Primary Key |
| ISBN | text | 20 | No | International Standard Book Number |
| name | text | 50 | No | Book Name |
| series | text | 20 | Yes | / |
| authors | text | 200 | Yes | / |
| press | text | 50 | Yes | / |
| size | text | 50 | Yes | Folio size |
| pages | num | int | Yes | / |
| price | num | float | No | / |
| introduction | text | 255 | Yes | Introduction of content |
| imgURL | text | 100 | Yes | Cover image URL |
| Publish date | text | 30 | Yes | / |
| Category | Text | 30 | Yes | Foreign Key |
| State | Num | Int | No | 0 for AVAILABLE, 1 for BORROWED, 2 for RENEWED, 3 for OVERDUE |

### 6.2.2 Category Table

The table is primarily designed to facilitate the management of library management , and query of books, librarians can set up different borrow period according to different types of books depending on the circumstances of the maintenance and management of books, the specific design of the structure of the table is as shown in Table 6.2.

Table 6.2 Category Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Whether allowed to be empty | Description |
| BookId | Text | 20 | No | Primary Key |
| type | Num | Int | Yes | / |
| period | Num | Int | Yes | Time period to borrow |
| Renew period | Num | Int | Yes | Time period to renew |
| dailyfine | Num | Float | Yes | Fined money per day when overdue |

### 6.2.3 User Table

User table is designed to manage the readers, readers need to enter information of the identity when borrowing books, the telephone and other information is for contacting with readers, reader type information determines the number of books one can borrow, and registration time can be used to check the identity of the reader to calculate the effective period. The specific structure of the table design shown in Table 6.3.

Table 6.3 User Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | | Field Length | Whether allowed to be empty | Description |
| id | text | 20 | | No | Primary Key | |
| Password | Text | 50 | | No | / | |
| name | Text | 20 | | Yes | User name | |
| gender | Num | Int | | Yes | 0 for male, 1 for female | |
| addresse | Text | 50 | | Yes | / | |
| Tel | Text | 20 | | Yes | Telephone | |
| startdate | Text | 50 | | Yes | Registration date | |
| enddate | Text | 50 | | Yes | / | |
| type | Num | Int | | Yes | 1 for undergraduates  2 for graduates  3 for teachers and PHDs | |

### 6.2.4 Dictionary Table

This table is designed in order to classify different users for the convenience of management, from the table we can tell readers from administrators. The table design is shown in Table 6.4.

Table 6.4 Dictionary Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Whether allowed to be empty | Description |
| userid | Text | 20 | No | Primary key |
| bookp | Num | Int | yes | Whether has the privilege of managing book (add, delete, modify) 0 for no, 1 for yes |
| readerp | Num | INT | yes | Whether has the privilege of managing readers(add, delete, modify) 0 for no, 1 for yes |
| parameterp | Num | Int | yes | Whether has the privilege of setting the parameters(fined money, borrow periods, borrow amount,etc.) 0 for no, 1 for yes |

### 6.2.5 Borrow Book Table

The design of the table is used to manage the readers to borrow books, barcode property is a unique identifier ID, ​​the ID number of records corresponding readers to borrow, borrow time records the corresponding borrow time, and due time is the return time according to the borrowing rule, the return time is the actual time user return the book, the table design is shown in Table 6.5.

Table 6.5 Borrow Book Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Whether allowed to be empty | Description | |
| id | Text | 20 | No | Id of borrow | |
| readerID | Text | 20 | No | Id of borrower |
| bookcode | Text | 20 | No | Id of book |
| borrowdate | Date | 20 | yes | / |
| duedate | Date | 20 | yes | / |
| returndate | Date | 20 | yes | / |
| renew | Num | Int | yes | 0 for no, 1 for renewed |

# 7 Development Process

## 7.1 Version 1.0 – Groundwork

**Goal**

The Goal for this iteration is creating a foundation for further development. It includes technology choosing, setting up framework and integrating with each other for whole system, and then basic functions of login service to make sure system work stable and well-connected.

**Details**

**Chosen Development technology and tools**

SSH framework



Figure 13 Division of tasks of SSH framework

**Integration of Spring framework and Hibernate framework**

It is easy to integrate Spring framework with Hibernate because of its extensibility and openness. Spring framework provides unified management of data source. Instead of configuring configuration file - **hibernate.cfg.xml** in the Hibernate, it only needs to configure data source and control attributes for the Hibernate in the **applicationContext.xml** file in the Spring. Meanwhile, in order to easy to use, Spring framework provides Hibernate Template which can easily control database without tedious work. In Spring, database connection and transaction management all begin with setting up SessionFactory. SessionFactory only requires one instance in the application, so the instance can be created by Spring and injected into related dependent objects. The Code for configuration file is as follows:

|  |
| --- |
| <!—configure Hibernate database source -->  <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close"  <!-- get database connection information from configuration file -->  p:driverClassName="${jdbc.driverClassName}"  p:url="${jdbc.url}"  p:username="${jdbc.username}"  p:password="${jdbc.password}" />  <!-- instantiate SessionFactory-->  <bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean"  <!-- reference data source -->  p:dataSource-ref="dataSource"  <!-- specify mapping files of Hibernate -->  p:mappingDirectoryLocations="classpath:/com/books/domain">  <property name="hibernateProperties"><props>  <!-- set the dialect of Hibernate -->  <prop key="hibernate.dialect">  org.hibernate.dialect.MysqlDialect  </prop>  <!-- background output SQL statements operated by Hibernate and format -->  prop key="hibernate.show\_sql">true</prop>  <prop key="hibernate.format\_sql">true</prop>  </props></property></bean>  <!-- configure HibernateTemplate -->  <bean id="hibernateTemplate"  class="org.springframework.orm.hibernate3.HibernateTemplate"  p:sessionFactory-ref="sessionFactory" /> |

Figure 14 Configure Hibernate into Spring

**Integration of Spring framework and Strus2 framework**

Spring not only offers an outstanding open source MVC framework referred as SpringMVC, but also supports well of integration with other web frameworks. During the integration of Spring and Struts2, the main step is making Spring’s IOC container manage Struts2’s Action, then the Action classes of Struts2 are able to require their instances through Spring. Before the integration, in Struts2’s configuration file - **struts.xml,** we need to convert Struts2’s request processor to Spring’s request processor, and upload ApplicationContext when starting Web. In addition, using comment @Controller can realize class injections for these Action classes need to be injected. Configuration code of web.xml is as follows:

|  |
| --- |
| <context-param>  <!—configure applicationContext.xml for files upload paths-->  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/applicationContext\*.xml</param-value>  </context-param>  <!—use ContextLoaderListener(from Spring) to create ApplicationContext-->  <listener>  <listener-class>  org.springframework.web.context.ContextLoaderListener  </listener-class></listener> |

Figure 15 Configure Spring into Struts2

**Tomcat server**

Tomcat server is a lightweight application server, and it is widely used to handle the normal amount of concurrent accesses in small or medium system. It is also the first choose for debugging JSP program. Moreover, Tomcat server has good compatibility and supported by many well-known software companies. It runs stable and has a good development prospect. Deployment diagram of Tomcat server for this system is as follows:



Figure 16 Deployment of Tomcat server

The following steps are the installation of Tomcat server in MyEclipse:

1. In MyEclipse, open display window from window – show view – servers, as shown in figure:



Figure 17 open window from the menu

2. Right click New – server in Servers window, it will let you to select version of Tomcat:



Figure 18 select version of Tomcat server

3. After done select version, click Next to access the place where to add directory for Tomcat server, then choose the proper directory and change JRE to JRE6, as shown in figure:



Figure 19 add directory for Tomcat server

4. After adding directory, click Next to the place where to upload project to Tomcat, choose the proper project and click Fish, as shown in figure:



Figure 20 upload the project to the server

**MySQL database**

MySQL is an open source relational database management system that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. Free-software-open source projects that require a full-featured database management system often use MySQL.

The following steps shows the installation of MySQL plugin in Eclipse:

1. open the Data Source Explorer window from window – show view – data Source Explorer:



Figure 21 Data source explorer window

2. Right click in Data Source Explorer windows, and open New Connection Profile window:



Figure 22 new connection profile window

3. In New Connection Profile window, select desire type of database for adding, and access to New Driver Definition window by double clicking. Then, configure all the values according to chosen database:



Figure 23 New driver definition window

4. Click finish to complete.

**Basic functions of login**

After setting up system frameworks and platforms, login in module will be the first priority to concern on, since realizing basic login module is very necessary for testing system frameworks configuration and integration, and I will explain the concept of SSH clearer by this practical instance. Login module services have two actors – normal user and administrator. Normal users can login and perform all the functions provided by the system to meet their demands, conversely, administrators perform their duties to manage system, they login the backstage of system to manage administration module.

Activity diagram of Login Account is illustrated by the following figure:



Figure 24 Activity diagram of Login process

Description

Actors: Normal User and Administrator

Actors access to login page, and input valid user name and password. System will verify User name and password, if is correct, login successfully, otherwise, login failed.

**Database design**

The following figure shows design of user table in database. All Users fell mainly into two classes: normal users and administrators, presented by user state 1 and user state 2 respectively. I also add certain attributes (Skype\_ID, email, etc.) related to further design of book-borrowing and living library system into the table.

User table of database shows as below:



Figure 25 User table of database

**Code Structure**

The system code structure belongs to three layers: Dao layer, Service layer and Action layer. Dao layer uses Hibernate to operate the underlying database; Action layer’s main job is controlling front web pages, passing parameters and calling Service layer to process business logic.

System code structure is showed as below:



Figure 26 Code structure

Since Dao layer uses Hibernate, it is easy to generate entity classes through the database table structure. It is unnessary to require mapping files from Hibernate becase of supportive comments of Java. Similarly, Struts2 and Spring also use comments, in this way, programmer can simplify work of configurate files and easily manage configurations.

The following code shows user name and password will be introduced to Action of Structs after their submits.

|  |
| --- |
| @Action(value = "login", results = {@Result(name = "success",type = "redirect", location = "/admin/main.jsp"),  @Result(name = "failure", location = "/admin/login.jsp")})  **public** String doLogin() {  List<User> users = userService.login(user);  **int** ret = users.size();  **if**(ret > 0) {  **return** "success";  } **else** {  **return** "failure";  }  } |

Action layer calls service layer to process user login logic, then Action decides actions for login successful or failed. This is how Struts2 plays the role here. Service layer calls Dao layer to operate database. After verification of information passed, administrators will be able to access to management interface of system backstage.

After successful login, administrators management interface is showed as below:



Figure 27 Administrator's home page

## 7.2 Version 1.1 – Basic Borrow

### 7.2.1 Search a book

To search a book by its title, ISBN, the press or the author, we implement this function in the BookService class, which uses BookDao to query book table in the database.

Take searching a book by its title for example, the code in the BookDao shows as below:

|  |
| --- |
| **final** String hql = "from Book b where b.name like :keyword";  List<Book> books = hibernateTemplate.executeFind(  **new** HibernateCallback<Object>() {  **public** Object doInHibernate(Session session) {  Query query = session.createQuery(hql);  query.setParameter("keyword","%"+keyword+"%");  query.setFirstResult(0);  query.setMaxResults(10);  List<Book> list = query.list();  **return** list;  }  }); |

Figure 28 Code Implementation for Searching a Book

The UI of searching a book is shown as below:

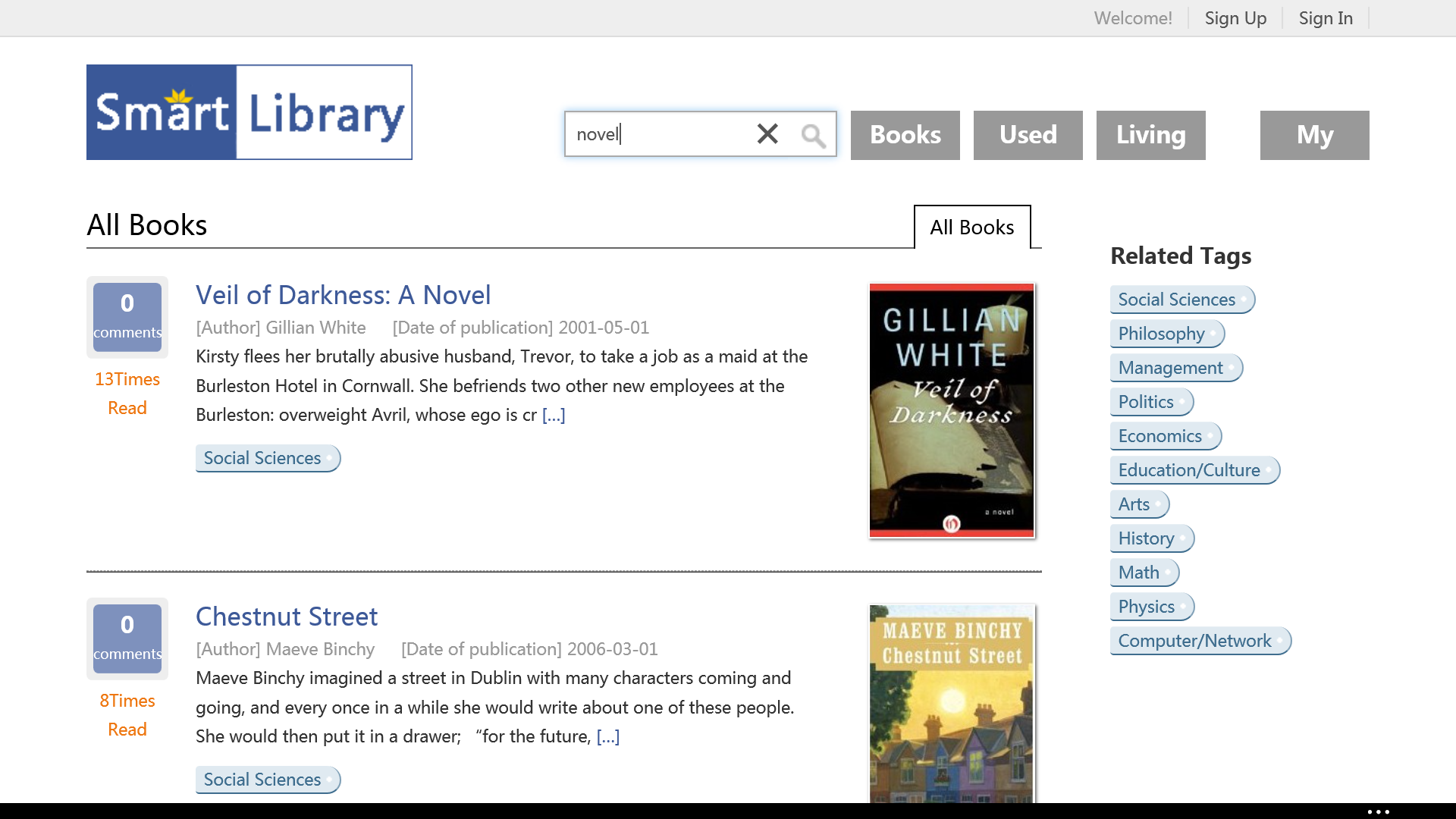


Figure 29 Search a book

### 7.2.2 User information

Users may modify personal information on the personal page, like password, telephone, address, etc. This is implemented in the UserService, which calls UserDao to access data, again, we illustrate this by the code in UserDao:

|  |
| --- |
| **public** **int** updateUser(User user) {  User u = hibernateTemplate.get(User.**class**, user.getId());  List list = hibernateTemplate.find("from TimeTable tt where user.id="+u.getId());  **if**(list.size()>0){  hibernateTemplate.deleteAll(list);  }  List<TimeTable> timeTables = user.getTimeTables();  u.setUsername(user.getUsername());  u.setRealname(user.getRealname());  u.setDept(user.getDept());  u.setSpecialty(user.getSpecialty());  u.setStudentId(user.getStudentId());  u.setGender(user.getGender());  u.setAge(user.getAge());  u.setPhone(user.getPhone());  u.setSkype(user.getSkype());  u.setEmail(user.getEmail());  u.setRemark(user.getRemark());  hibernateTemplate.update(u);  **for** (**int** i = 0; i < timeTables.size() ; i++) {  TimeTable timeTable = timeTables.get(i);  timeTable.setUser(u);  timeTable.setState(1);  hibernateTemplate.save(timeTable);  }  **return** 1;  } |

Figure 30 Code Implementation for User Modifying Information

The UI of this page shows as:

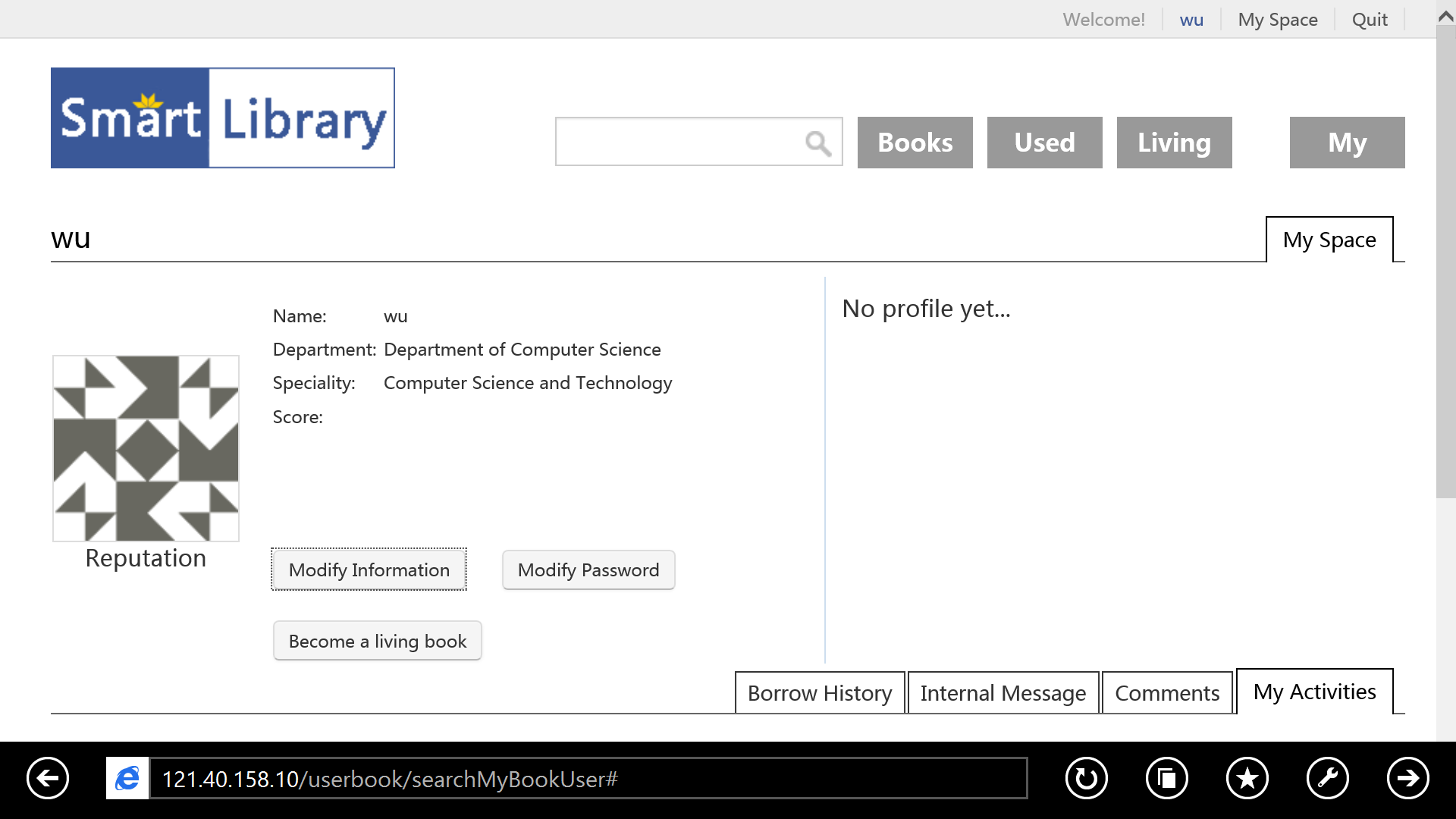


Figure 31 User Information Page

### 7.2.3 Search a living book

Mainly we support searching a living book by its name at this phrase, this is implemented in the class UserBookService and UserBookDao. The code that works goes like this:

|  |
| --- |
| **public** List<User> getUserBookList(User user,String keyword, **final** Integer start) {  StringBuffer sql = **new** StringBuffer()  .append("from User u ")  .append("where u.state > 0 and u.flag=1");  **if**(user!=**null** && user.getSpecialty()!=**null** && !user.getSpecialty().equals("")) {  sql.append("and u.id != " + user.getId());  // sql.append(" and u.specialty.id = " + user.getSpecialty().getId());  };  **if** (keyword!=**null** && !keyword.equals("")) {  sql.append(" and u.specialty.name like '%" + keyword + "%' ");  sql.append(" or u.username like '%" + keyword + "%' ");  }  **final** String hql = sql.toString();  List users = hibernateTemplate.executeFind(**new** HibernateCallback<Object>() {  **public** Object doInHibernate(Session session) {  Query query = session.createQuery(hql);  query.setFirstResult(start);  query.setMaxResults(12);  List list = query.list();  **return** list;  }  });  **return** users;  } |

Figure 32 Code Implementation for Searching a living book

The UI of searching a living book is in some way integrated with searching a book for better UI experience, this is how it looks like:

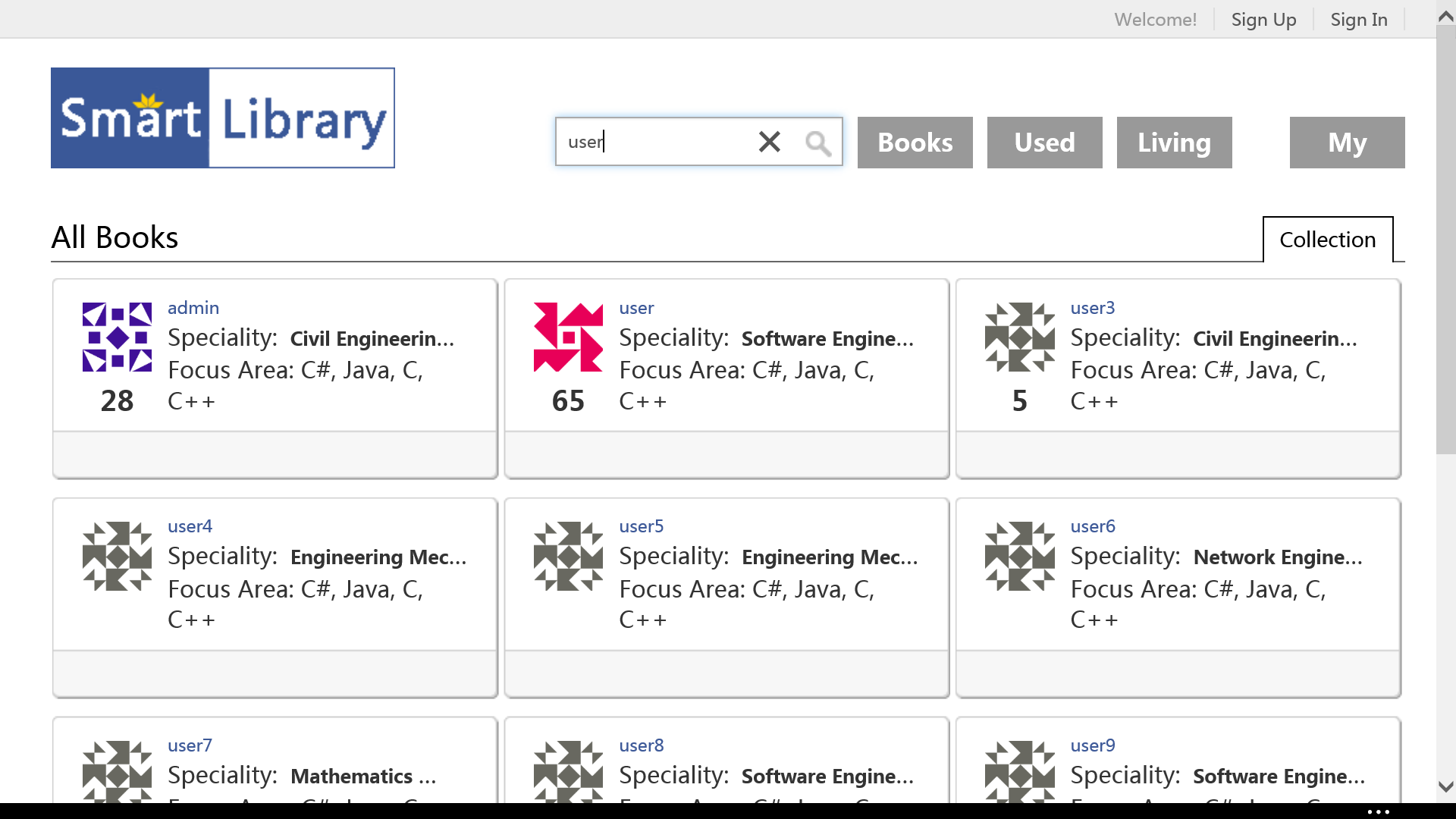


Figure 33 Search a living book

### 7.2.4 Borrow, return and renew

The system feature of borrowing a book provides readers to borrow a book with their identity number and barcode of the book from librarians, and a borrowing needs the librarians to confirm, once borrowed successfully, the book state is set to BORROWED. So is the returning of a book, when confirmed by librarians, the state of the book in the database changes back to AVAILBLE, waiting to be borrowed by other readers.

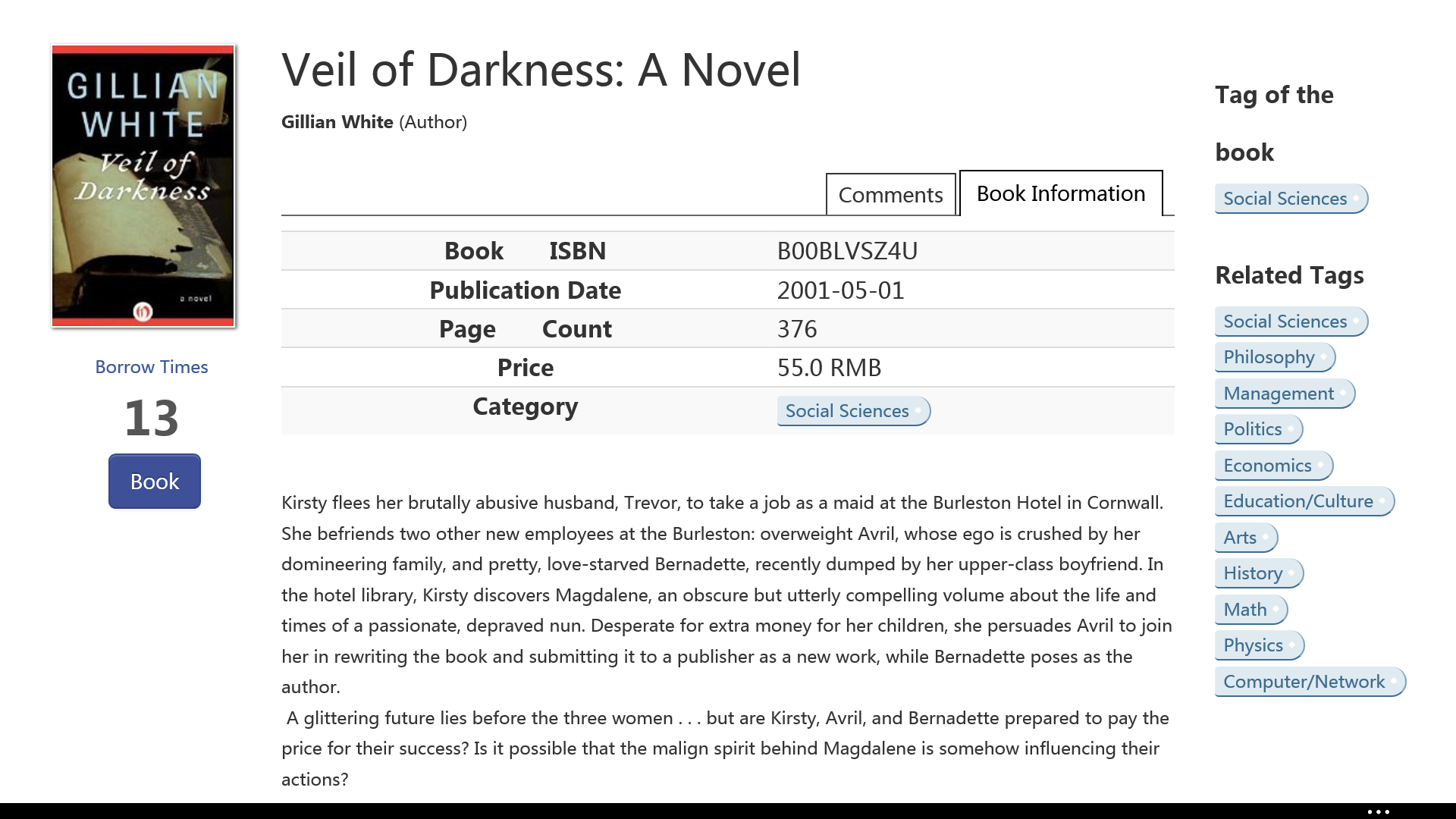


Figure 34 To borrow a book, click the Book button

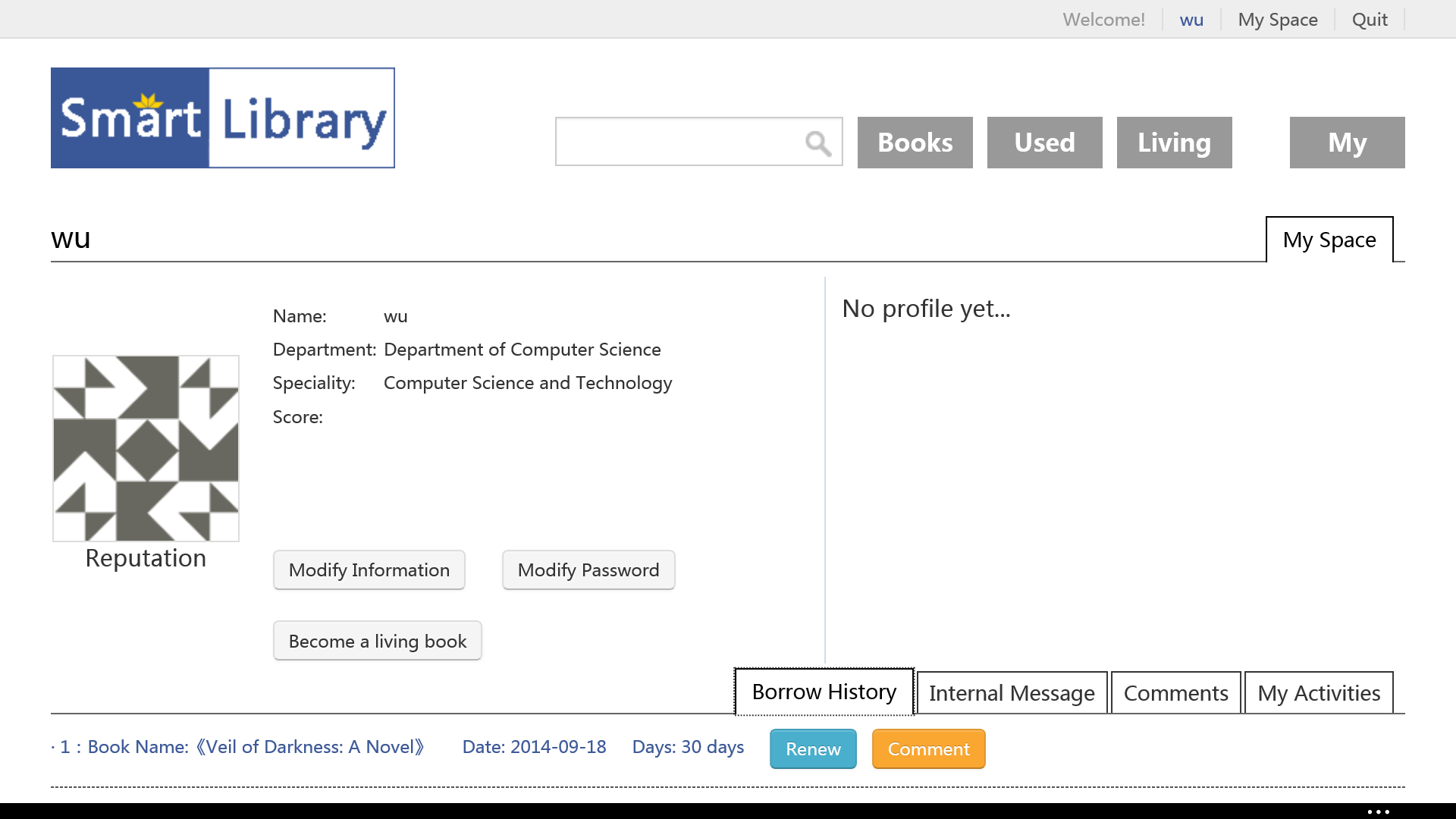


Figure 35 One record appears in Borrow History when complete borrowing a book

The code of borrowing is shown as below:

|  |
| --- |
| if (judge.judgeBookBorrow(book) && judge.judgeReaderBorrow(book)  && judge.judgeBookYesOrNot(book) && judge.endDate(book) == true) {  if (judge.readerType(readerForm) == 1  && judge.bookCount(readerForm) >=judge.bookMaxNumber(readerForm)) {  JOptionPane.showMessageDialog(null, "Undergraduates can borrow at most "+judge.bookMaxNumber(readerForm)+"!", "Warning",JOptionPane.WARNING\_MESSAGE);  }  if (i == JOptionPane.YES\_OPTION) {  LibDataAccessor b = new LibDataAccessor();  b.borrowBookInfo(book,judge.dayMaxNumber(readerForm));//  int n = b.readerBorrowBook(book,judge.dayMaxNumber(readerForm));  if (n > 0) {  int j = JOptionPane.showConfirmDialog(null,"Borrow successfully!",  JOptionPane.WARNING\_MESSAGE);  if (j == JOptionPane.YES\_OPTION) {  bookFieldText.setText("");  } else {  dispose();  } |

Figure 36 Code Implementation for Borrowing a book

The code of returning is shown as below, firstly we need to check if the book is existed, if the book is borrowed, if the book is renewed, and if the book is overdue.

|  |
| --- |
| if (b.judgeBookBorrow(book) && b.judgeBookYesOrNot(book) == false) {  int overduedays = b.overDueDays(book);// get the overdue days  int renew = b.renew(book);// check if renewed  float money1 = b.moneyUpdate(book,moneyNumber);// if overdue, get the fined money  String money = b.numberFormatter(money1);  int i = JOptionPane.showConfirmDialog(null, "Confirm returning?", "Yes",  JOptionPane.YES\_NO\_OPTION);  if (i == JOptionPane.YES\_OPTION) {  if (overduedays < 0) {  b.returnBookInfo(book); b.returnBookLendInfo(book); int j = JOptionPane.showConfirmDialog(null, "Return successfully! Continue?","Return successfully", JOptionPane.WARNING\_MESSAGE);  if (j == JOptionPane.YES\_OPTION) {  bookFieldText.setText("");} else {  dispose();}}  else if (overduedays == 0) {  int a = JOptionPane.showConfirmDialog(null, "The book is going to be overdue. Renew?", JOptionPane.WARNING\_MESSAGE);  if (a == JOptionPane.YES\_OPTION) {  boolean flag = b.renewBook(book); // renew the book  b.returnBookInfo(book);  b.returnBookLendInfo(book);  if (m == JOptionPane.YES\_OPTION) {  bookFieldText.setText("");  } else {dispose();} |

Figure 37 Code Implementation of Returning and Renewing

### 7.2.5 Used book trade

Used book trade platform allows users to post information about their used book for sell or purchase. The page looks like:

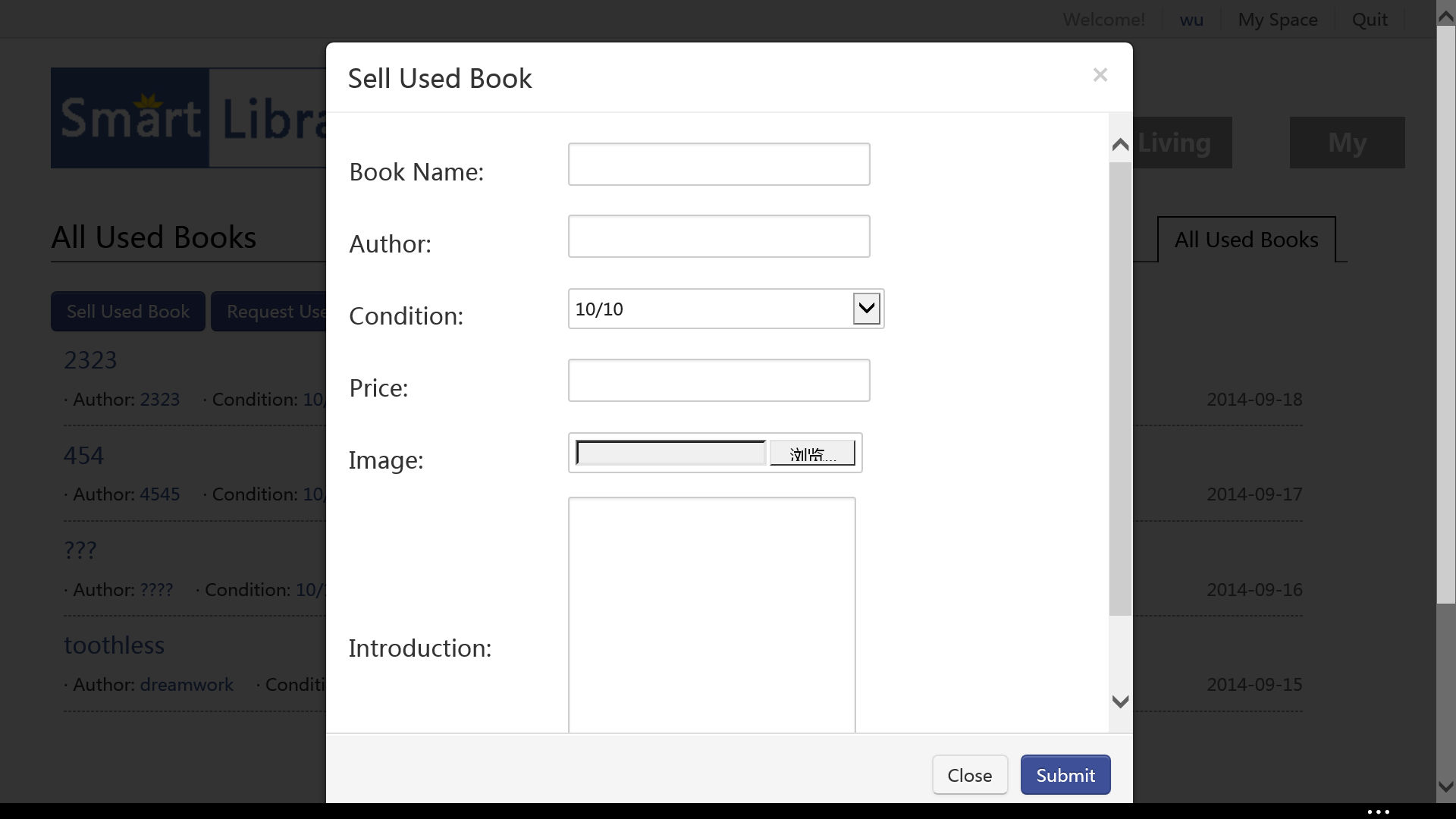


Figure 38 Used book trade

The code is rather straight and simple, take a glimpse at the UsedBookService, it’s what we provide in this module.

|  |
| --- |
| **public** **class** UsedBookService {  @Autowired  **private** UsedBookDao usedBookDao;  **public** **int** saveUsedBook(UsedBook usedBook) {  **return** usedBookDao.saveUsedBook(usedBook);  }  **public** List<UsedBook> queryUsedBooks(String keyword) {  **return** usedBookDao.queryUsedBook(keyword);  }  **public** UsedBook acquireUsedBook(UsedBook usedbook) {  **return** usedBookDao.acquireUsedBook(usedbook);  }  **public** **void** saveComment(UsedBookComment usedBookComment) {  usedBookDao.saveComment(usedBookComment);  }  **public** List<UsedBookComment> queryUsedBookComment(UsedBook ub) {  **return** usedBookDao.queryUsedBookComment(ub);  }  **public** List<UsedBook> queryUsedBookList(**int** start, **int** limit) {  **return** usedBookDao.queryUsedBookList(start,limit);  }  **public** **int** getTotalCount() {  **return** usedBookDao.getTotalCount();  }  **public** **int** deleteUsedBook(Long usedBookId) {  **return** usedBookDao.deleteUsedBook(usedBookId);  }  } |

Figure 39 Code Implementation of Used Book Service

## 7.3 Version 1.2 – Ranking and Recommendation

As for the ranking, we use HQL sentence to sort data, which is based on Hibernate. For example, we need to rank the top borrowed books, well the Book has ID and the borrow count property, we can sort the book by this property in a descending order.

|  |
| --- |
| Query q=session.createQuery("from Book order by borrowedCount desc");  List<Book> li=(List<Book>)q.list();  **for** (**int** i = 0; i < li.size(); i++) {  System.out.println(li.get(i).getName());  }  **return** li; |

Figure 40 Ranking Top Borrowed Books

The ranking results shows on the home page like:



Figure 41 Ranking Result

As we can see from the figure, there’s recommend reading on the left, this recommend list is formed by the users’ rating score, besides this, we have another recommendation shown as below:

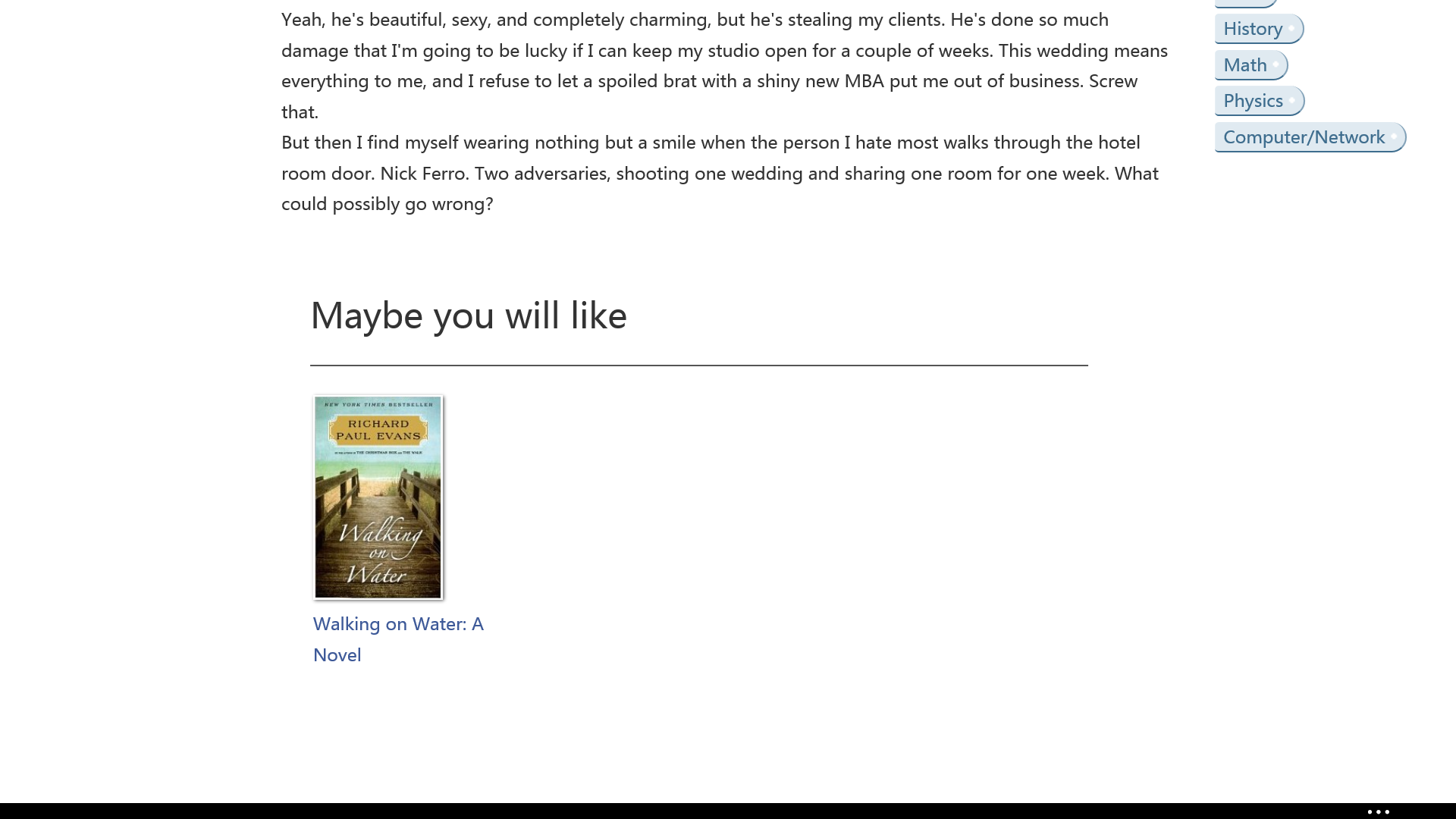


Figure 42 Maybe you will like

Well, this kind of recommendation is a bit complicated, ‘cause we need to find relationships between books, one kind of relationships could be the same tags, another is based on the user actions, most of the users who borrowed this book also borrow that book, so we recommend that book. We implement this algorithm in a simple and straight way, RecommendAction will call the BorrowBookService to get recommending list to show on the page, the BorrowBookService will call the algorithm code in the BorrowBookDao, the code is shown as below:

|  |
| --- |
| @Override  **public** List<Book> getSupportRatingBooks() {  List<String> bookSet = **this**.makeBookSet();  **int** index = 0;    Book[] books = **new** Book[bookSet.size()];  List<Book> resultBookThree = **new** ArrayList<Book>();  Session session = **null**;  **try**{  Configuration config = **new** Configuration().configure();  @SuppressWarnings("deprecation")  SessionFactory sessionFactory = config.buildSessionFactory();  session = sessionFactory.openSession();    **for**(Iterator<String> iter = bookSet.iterator(); iter.hasNext();){  Books[index] = **new** Book();  books[index].setBookIds(iter.next());    **long** support = (Long)session.createQuery("select count(\*) from com.books.domain.BorrowBook as b where b.bids like '%"+books[index].getBookIds()+"%'").list().get(0);  // .setParameter("bookId", "%"+books[index].getBookIds()+"%")    books[index].setSupportRating((**int**)support);  index++;  }    **for**(**int** i = 0; i<books.length; i++){  System.out.println(books[i].getBookIds());  System.out.println(books[i].getSupportRating());  }  //  **int** num\_one = 0;  **for**(**int** i = 0; i < books.length; i++){  **if**(books[i].getSupportRating() > 1){  books[num\_one++] = books[i];  }  }  //  **int** length = (num\_one\*(num\_one-1))/2;  **int** tempIndex = 0;    Book[] bookTwo = **new** Book[length];    **for**(**int** i = 0; i < num\_one; i++){  **for**(**int** k = i+1; k < num\_one; k++){  StringBuffer sb = **new** StringBuffer();  sb.append(books[i].getBookIds()).append(',').append(books[k].getBookIds());    bookTwo[tempIndex] = **new** Book();  bookTwo[tempIndex].setBookIds(sb.toString());  bookTwo[tempIndex].setSupportRating(0);    tempIndex++;  }  }  //  **for**(**int** i = 0; i < bookTwo.length; i++){  String[] bookIds = bookTwo[i].getBookIds().split(",",2);    **long** support = (Long)session.createQuery("select count(\*) from com.books.domain.BorrowBook as b where b.bids like '%"+bookIds[0]+"%' and b.bids like '%"+bookIds[1]+"%'").list().get(0);  // .setParameter("bookId1", "%" + bookIds[0] +"%")  // .setParameter("bookId2", "%"+ bookIds[1] + "%")  // .uniqueResult();  bookTwo[i].setSupportRating((**int**)support);  }    **for**(**int** i = 0; i<bookTwo.length; i++){  System.out.println(bookTwo[i].getBookIds());  System.out.println(bookTwo[i].getSupportRating());  }  //  **int** index\_two = 0;  **for**(**int** i = 0; i < bookTwo.length; i++){  **if**(bookTwo[i].getSupportRating() > 1){  bookTwo[index\_two++] = bookTwo[i];  }  }    List<Book> tempBookThree = **new** ArrayList<Book>();  **for**(**int** i = 0; i < index\_two; i++){  **for**(**int** k = i+1; k < index\_two; k++){  String first = bookTwo[i].getBookIds().substring(0, bookTwo[i].getBookIds().indexOf(','));  String end = bookTwo[k].getBookIds().substring(0,bookTwo[k].getBookIds().indexOf(','));  **if**(first.equals(end)){  Book temp = **new** Book();  StringBuffer sb = **new** StringBuffer();  sb.append(bookTwo[i].getBookIds()).append(',').append(bookTwo[k].getBookIds().substring(bookTwo[k].getBookIds().indexOf(',')+1));    temp.setBookIds(sb.toString());  temp.setSupportRating(0);  tempBookThree.add(temp);  }    }  }    **for**(**int** i = 0; i<tempBookThree.size(); i++){  System.out.println(tempBookThree.get(i).getBookIds());  System.out.println(tempBookThree.get(i).getSupportRating());  }    Book maxSupportRatingBook = **null**;  **int** maxSupportRating = -1;  **for**(Book book: tempBookThree){  String[] bookIds = book.getBookIds().split(",",3);  **long** support = (Long)session.createQuery("select count(\*) from com.books.domain.BorrowBook b where b.bids like '%"+bookIds[0]+"%' and b.bids like '%"+bookIds[1]+"%' and b.bids like '%"+bookIds[2]+"%'").list().get(0);  // .setParameter("bookId1", "%"+bookIds[0]+"%")  // .setParameter("bookId2", "%"+bookIds[1]+"%")  // .setParameter("bookId3", "%"+bookIds[2]+"%")  // .uniqueResult();    book.setSupportRating((**int**)support);  // if(support > maxSupportRating){  // maxSupportRating = (int)support;  // maxSupportRatingBook = book;  // }  }      **for**(**int** i = 0; i < tempBookThree.size(); i++){  **if**(tempBookThree.get(i).getSupportRating() > 1){  resultBookThree.add(tempBookThree.get(i));  System.out.println("Recommend Result: "+tempBookThree.get(i).getBookIds()+"Support Rating: "+tempBookThree.get(i).getSupportRating());  }  }  // System.out.println("Set: "+maxSupportRatingBook.getBookIds()+"," +" SupportRating: " + maxSupportRatingBook.getSupportRating());    }**catch**(Exception e){  e.printStackTrace();    }**finally**{  **if**(session.isOpen()){  session.close();  }  }  **return** resultBookThree;  } |

Figure 43 Code Implementation for Recommendation Algorithm

## 7.4 Version 1.3 – Notification

Notification system, as the name suggests, it means notifying and conveying information. In our library management system, we use notification system to notify users their reservation of the living book and borrowed books to be overdue by emails at this phrase. We aim to allow users to get the news and needs to be reminded and processed timely. So maybe in the future, we will implement the function that notifies users about newest books which they are interested in, comments received, warnings given by administrators, and so on, and the form may vary, using wechat or other Medias to notify users is the future enhancement.

However, it’s easy to say, hard to implement. We make a research in this area, like what we did in implementing the recommendation, and concluded that the notification system design principles can be simply summarized as follows:

**The message should spread with highest efficiency** (acquisition, processing, information communication, user feedback efficiency, etc.)

**Avoid harassment** (noise, frequent tips)

The core logic goes as:

Notifications need to be merged before pushed, in order to improve the efficiency of information dissemination and reduce harassment, noise reduction, what’s more, to balance servers’ stress.   
1) The merged cycle: All news summarize within a fixed time (within 24 hours / 30 days, etc.);   
No fixed time (as long as untreated / unread) Of course, generally we merge in a combination with the two forms: merge unprocessed messages within 24 hours  
2) Classification  
Merge the same type   
Merge the same sponsor (e.g. Joe Smith send you the n-th private letter)   
Merge the same time period (e.g. 24 hours received a total of n reviews)

According to the previously mentioned method of distribution for processing notifications, it can be logically divided into two layers: process the notification of content and the notification of states. A narrow understanding of the state is that the notification has been read (processed).   
The initial number is usually the total unread messages pushed by the system, if the user clicks on a list of figures-related functions and check the messages, then the state of the message changes to read, so the number will be reduced.

While some important higher-level message can be defined to change state only after the user take certain actions about that message. For example, user comments, only an action of replying, or clicking Ignore or clicking delete can change that message into the state of read.

A narrow understanding about processing contents is that whether the user do the operations.   
According to the needs of different kinds of messages and operations, the operation can be divided into:   
**Processing**: The user must click the link function for processing. Such as: Your password is too simple, click here to be modified;   
**Reply**: If the user replies a private letter of comment;   
**Confirmation**: feedback on the message to make sure whether the user has noticed, for example some system prompts a dialog with "I already know, not prompt" option;   
**Ignore**: The user does not carry out any operation or operations;   
**Delete**: user deletes this message.

The callback of notification is mainly for the processed message. Trigger messages between users generally need to be kept on file saver such as comment / reply / private letters. The system offers the option to ask the user if execute a certain period of automatic cleanup. In some situations, we also need to consider the priority functions. System messages are generally set up to trigger the callback system at a certain time. Automatically delete the expired messages (user is not logged on for a long time but received replies of others) or take some actions according to business needs. Failed to read the private letters / comments / replies permanently retained and so on. Important unread messages may try to push or use other means (mail, APP, wechat, etc.) notification.

Currently we use more often the current trigger and immediate action similar to the "WYSIWYG" interaction. The reasons to use this approach mainly are: Message notification is put at the global navigation of the page, so that we can receive the new messages timely when we are browsing any panels on the page. After the completion of message processing in a floating layer, the user can proceed the operation without disturbance. Due to the limited navigation area, there’s a need for the same kind of messages to be unified (Facebook classifies as friend request, private letters, notifications).Provide historical records. Mark read or unread states. Anti-harassment (disturb), too many useless notification is bound to cause noise to disturb the user. Therefore, we should set reasonable frequencies and channels of notification messages to prevent the loss of experience and efficiency.

Since we use SSH framework, we look up for some developed framework to help us achieve this mission of notification. So far as we explored, we find that the Spring Framework can choose three kinds of different timing mechanisms.

**TimerTask-based mechanisms**

The Java TimerTask class can be used to perform regular tasks, which can be called by the Spring framework package. Where the execution cycle ScheduledTimerTask class definition task, timerTask attribute specifies the working class execution; TimerFactoryBean class is responsible for starting regular tasks, scheduledTimerTasks property displays a list of tasks you need to start the timer. TimerTask suitable for a relatively short time interval simple task.

**Quartz-based mechanisms**   
Quartz is an open source task scheduling framework that provides a more powerful than TimerTask task scheduling execution of enterprise-class features, Spring inherited and simplify it. Quartz has great flexibility without losing the simplicity that can perform complex tasks scheduling. It allows developers to define flexible scheduling rules triggers, and triggers and tasks can be associated mappings. Quartz provides a convenient class for Spring-based applications, such as MethodlnvokingJobDetailFactoryBean, SimpleTriggerBean, CronTriggerBean and SchedulerFactoryBean.

**Executor-based mechanisms**

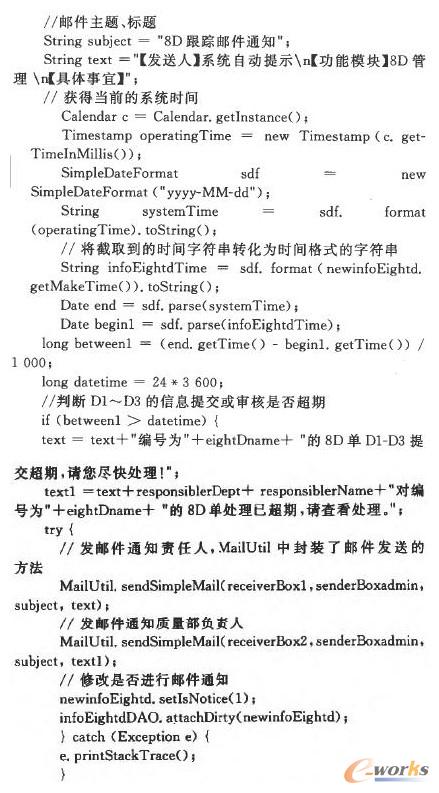
Executor framework by the thread pool to handle asynchronous tasks, you can also perform the task scheduling service. Rules to specify which ScheduledExecutorTask class scheduling, ScheduledExecutorFactoryBean class to complete scheduled tasks, and can use it to control the number of threads in the thread pool.

Based on the above analysis, TimerTask simple timed to meet business needs; Quartz task scheduler services offered the most complete, in a very wide range of enterprise applications; Executor thread pool to provide the most perfect service. Below with Spring framework to implement the use of Quartz timer design.

Timer framework design   
Quartz timer-based framework must first import the required jar package, and then it provides some classes according to successively achieve the appropriate configuration. Detailed steps are as follows:   
     (1) declare the work class that must implement appropriate business logic, while the class dao object can also be included;   
     (2) create a task scheduling by MethodInvokingJobDetailFactoryBean class and the working class need to specify the tasks and implement business logic methods;   
     (3) two class provided by Spring to achieve the task of scheduling rules, they specifically: SimpleTriggerBean class configuration simple trigger (scheduling) rules, timing and frequency specified in task execution; CronTriggerBean trigger custom class configuration mode (scheduling) rule, it features than SimpleTriggerBean achieve the function to be strong, able to control the task trigger (scheduling) the exact time needed to specify scheduling final mapping between the trigger and the task;   
    (4) Finally, start the trigger by SchedulerFactoryBean.

Development timer framework can be attributed to 2 parts: Writing task classes and write configuration files.

**writing task class**   
In com. cssrc. quartz. Under the new service path to a JobService class by the class of methods to achieve tasks such as voidautoSendEmail (), the method used to send messages to perform scheduled tasks. Dao first need to declare the method used to send messages, such as userDAO, roleDAO, infoEightdDAO, infoOneThreedDAO and infoFourFivedDAO etc., and generate the corresponding getter and setter methods.

[[http://articles.e-works.net.cn/articles/images/zoom.gif](http://articles.e-works.net.cn/articles/articleimage/20126/129846471615000000_new.jpg)](http://articles.e-works.net.cn/articles/articleimage/20126/129846471615000000_new.jpg)

autoSendEmail methods to judge asked for this code first time the current system time and 8D information table created is more than l day, if exceeded, send regular mail to the responsible person by MailUtil class sendSim-pleMail method need to provide four parameters data, they were receiving mail, sending mail, e-mail subject and content. .

**write the configure file**

Create a new configure file applicationquartz.xml under the directory of WEB-INF, and this file must be stated in the web.xml of the project. The concrete configure file goes as below:

[[http://articles.e-works.net.cn/articles/images/zoom.gif](http://articles.e-works.net.cn/articles/articleimage/20126/129846472156093750_new.jpg)](http://articles.e-works.net.cn/articles/articleimage/20126/129846472156093750_new.jpg)

This profile describes in detail the quartz framework trigger sendEmail task after the system startup in 24 h, autoSendEmail method is specified by the task, quartz tag JobService class is the quality management that can achieve mail tasks, because in every 24h, it is triggered, so it can achieve with an e-mail reminder under the system timer framework.

## 7.5 Version 2.0 – Login with QR code

Let’s picture a scene first.

It’s Monday morning, your head is quite painful after last night’s party that finished at 5am, and your desktop computer died on Friday afternoon, while you were trying to do as much as possible, before the deadline that has been already moved twice. Bummer.

The broken computer has been replaced by your roommate’s private laptop, which by all chance has thousands of suspicious apps, key loggers and its running Vista. Ugh… Now, how do you log into your library management system without risking that your passwords will be stolen and your accounts compromised?

**Here comes the QR-logging-in idea:**

The Library management system is displaying QR code on the login screen, right next to regular username/password login form. You’re taking out your phone, snapping a photo with any QR reader app for your iPhone/Android device and in less than 5 seconds you’re logged into the website on the computer**.** No passwords, no hassle. And you didn’t even have to touch the keyboard!

**So, how to achieve that?**

The main goal was to avoid typing in anything and let users log in from any machine, even the one they haven’t used before, having only their phone within easy reach.

We are assuming that the phone has built-in camera, installed QR reader app, a browser capable of saving cookies and some internet connection (wifi/3g/gprs/whatever). That fits most iPhone/Android/Windows Phone/Blackberry users out there.

As previously mentioned, the web app is showing QR code. The code contains address to the central server + unique identifier of the token that has been assigned by server to the session on the desktop computer. After scanning the code, the phone is opening a page that checks if it has been used with this service before by looking for a cookie containing encrypted information about the user’s credentials (hash of username/user id). The hash is being checked against server’s database and if it’s valid - token in database is being updated with information that the access is granted to user X. Phone shows information that the user has logged to site XYZ. Browser on desktop is constantly checking status of the token and once it says that the user has “logged in” - it’s redirecting to secure part of website. Job done!

If the phone is not paired with any user account - the desktop computer will show regular log in form, so it would be good idea to log in first time from machine that you trust. After filling out the log-in form - user can log in automatically, as described above. Want to make it safer (but less convenient)? Log in form for first time users could be displayed on the phone instead.

This idea could probably work much better than Open ID/Browser ID. Multiple websites could use one centralized system that would allow user to pair his /her phone once and use it everywhere, without typing anything anywhere. No more typos, no more forgotten username/passwords.

Here’s how we implement QR login:

|  |
| --- |
| **package** com.books.wechat.action;  **import** com.books.dao.UserDao;  **import** com.books.domain.BorrowBook;  **import** com.books.domain.BorrowUser;  **import** com.books.domain.User;  **import** com.books.domain.UserComment;  **import** com.books.service.BorrowBookService;  **import** com.books.service.UserBookService;  **import** com.books.service.UserService;  **import** com.books.service.UtilService;  **import** com.books.wechat.bean.UserMap;  **import** com.books.wechat.util.PropUtil;  **import** com.books.wechat.util.WeixinUtil;  **import** com.opensymphony.xwork2.ActionContext;  **import** org.apache.struts2.ServletActionContext;  **import** org.apache.struts2.convention.annotation.Action;  **import** org.apache.struts2.convention.annotation.Namespace;  **import** org.apache.struts2.convention.annotation.ParentPackage;  **import** org.apache.struts2.convention.annotation.Result;  **import** org.springframework.beans.factory.BeanFactory;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.context.ApplicationContext;  **import** org.springframework.context.support.ClassPathXmlApplicationContext;  **import** org.springframework.context.support.FileSystemXmlApplicationContext;  **import** org.springframework.stereotype.Controller;  **import** javax.servlet.http.HttpServletResponse;  **import** java.io.PrintWriter;  **import** java.util.List;  **import** java.util.Map;  @Controller  @ParentPackage("default-package")  @Namespace("/")  **public** **class** LoginAction {  **private** String openid;  **private** String username;  **private** String password;  **private** String scene\_id;  @Autowired  **private** UserService userService;  @Autowired  **private** UserBookService userBookService;  @Autowired  **private** BorrowBookService borrowBookService;  @Action(value = "showQRCode", results = @Result(name = "json\_success", type = "json"))  **public** String showQRCode() **throws** Exception {  HttpServletResponse response = ServletActionContext.getResponse();  response.setContentType("text/plain;charset=UTF-8");  PrintWriter out = response.getWriter();  String scene\_id = WeixinUtil.*get9Num*();  WeixinUtil.*createScene*("QR\_SCENE",scene\_id , PropUtil.*get*("appid"),PropUtil.*get*("secret"));  **if**(**null**!=WeixinUtil.*QRCode*){  out.print("{'success':'true','QRCode':'"+ WeixinUtil.*QRCode* +"','scene\_id':'"+scene\_id+"'}");  System.out.println("{QRCode:"+ WeixinUtil.*QRCode* +",scene\_id:"+scene\_id+"}");  }  **return** **null**;  }  @Action(value = "bindWechat", results = @Result(name = "json\_success", type = "json"))  **public** String bindWechat() **throws** Exception {  HttpServletResponse response = ServletActionContext.getResponse();  response.setContentType("text/plain;charset=UTF-8");  PrintWriter out = response.getWriter();  User user = **new** User();  user.setUsername(username);  user.setPassword(password);  user.setOpenid(openid);  List<User> users = userService.login(user);  **if** (users == **null** || users.size()==0) {  out.print("{'success':'false'}");  } **else** {  User u = users.get(0);  u.setOpenid(user.getOpenid());  userService.alterUser(u);  }  out.print("{'success':'true'}");  **return** **null**;  }  @Action(value = "wechatLogin", results = @Result(name = "json\_success", type = "json"))  **public** String wechatLogin() **throws** Exception {  HttpServletResponse response = ServletActionContext.getResponse();  response.setContentType("text/plain;charset=UTF-8");  PrintWriter out = response.getWriter();  **if** (scene\_id != **null** && !scene\_id.equals("")) {  UserMap userMap = UserMap.*getInstance*();  String key = "qrscene\_" + scene\_id;  **if**(userMap.containsKey(key)){  out.print("{'success':'true','scene\_id':'"+ scene\_id +"'}");  } **else** {  out.print("{'success':'false'}");  }  } **else** {  out.print("{'success':'false'}");  }  **return** **null**;  }  **public** String getOpenid() {  **return** openid;  }  **public** **void** setOpenid(String openid) {  **this**.openid = openid;  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  **public** String getScene\_id() {  **return** scene\_id;  }  **public** **void** setScene\_id(String scene\_id) {  **this**.scene\_id = scene\_id;  }  } |

Figure 44 Code Implementation for QR login

While we need the support of Wechat Utility, which is shown as below:

|  |
| --- |
| **package** com.books.wechat.util;  **import** java.io.BufferedReader;  **import** java.io.File;  **import** java.io.FileReader;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.InputStreamReader;  **import** java.io.OutputStream;  **import** java.net.ConnectException;  **import** java.net.URL;  **import** java.util.Random;  **import** javax.net.ssl.HttpsURLConnection;  **import** javax.net.ssl.SSLContext;  **import** javax.net.ssl.SSLSocketFactory;  **import** javax.net.ssl.TrustManager;  **import** net.sf.json.JSONObject;  **public** **class** WeixinUtil {  **public** **final** **static** String *access\_token\_url* = "https://api.weixin.qq.com/cgi-bin/token?grant\_type=client\_credential&appid=APPID&secret=APPSECRET";    **public** **final** **static** String *user\_info\_url* = "https://api.weixin.qq.com/cgi-bin/user/info?access\_token=ACCESS\_TOKEN&openid=OPENID&lang=zh\_CN";    **public** **final** **static** String *scene\_url* = "https://api.weixin.qq.com/cgi-bin/qrcode/create?access\_token=TOKEN";    **private** String q\_appid = **null**;  **private** String q\_appsecret = **null**;  **public** **static** String *QRCode* = **null**;  **public** **static** JSONObject httpRequest(String requestUrl, String requestMethod, String outputStr) {  JSONObject jsonObject = **null**;  StringBuffer buffer = **new** StringBuffer();  **try** {  TrustManager[] tm = { **new** MyX509TrustManager() };  SSLContext sslContext = SSLContext.getInstance("SSL", "SunJSSE");  sslContext.init(**null**, tm, **new** java.security.SecureRandom());  SSLSocketFactory ssf = sslContext.getSocketFactory();  URL url = **new** URL(requestUrl);  HttpsURLConnection httpUrlConn = (HttpsURLConnection) url.openConnection();  httpUrlConn.setSSLSocketFactory(ssf);  httpUrlConn.setDoOutput(**true**);  httpUrlConn.setDoInput(**true**);  httpUrlConn.setUseCaches(**false**);  httpUrlConn.setRequestMethod(requestMethod);  **if** ("GET".equalsIgnoreCase(requestMethod))  httpUrlConn.connect();  **if** (**null** != outputStr) {  OutputStream outputStream = httpUrlConn.getOutputStream();  outputStream.write(outputStr.getBytes("UTF-8"));  outputStream.close();  }  InputStream inputStream = httpUrlConn.getInputStream();  InputStreamReader inputStreamReader = **new** InputStreamReader(inputStream, "utf-8");  BufferedReader bufferedReader = **new** BufferedReader(inputStreamReader);  String str = **null**;  **while** ((str = bufferedReader.readLine()) != **null**) {  buffer.append(str);  }  bufferedReader.close();  inputStreamReader.close();  inputStream.close();  inputStream = **null**;  httpUrlConn.disconnect();  jsonObject = JSONObject.fromObject(buffer.toString());  } **catch** (Exception e) {  }  **return** jsonObject;  }  **public** **static** AccessToken getAccessToken(String appid, String appsecret) {  AccessToken accessToken = AccessToken.*getInstance*();  String requestUrl = *access\_token\_url*.replace("APPID", appid).replace("APPSECRET", appsecret);  JSONObject jsonObject = *httpRequest*(requestUrl, "GET", **null**);  **if** (**null** != jsonObject) {  **try** {  accessToken.setToken(jsonObject.getString("access\_token"));  accessToken.setExpiresIn(jsonObject.getInt("expires\_in"));  } **catch** (Exception e) {  accessToken = **null**;  }  }  **return** accessToken;  }  **public** **static** String createScene(String action\_name,String sceneId,String appid, String appsecret) {    String str = "{\"expire\_seconds\": 1800,\"action\_name\":\"QR\_SCENE\", \"action\_info\": {\"scene\": {\"scene\_id\":\""+sceneId+"\"}}}";  **if**(action\_name.equals("QR\_LIMIT\_SCENE")) {  str = "{\"action\_name\":QR\_LIMIT\_SCENE, \"action\_info\": {\"scene\": {\"scene\_id\":"+sceneId+"}}}";  }  AccessToken accessToken = WeixinUtil.*getAccessToken*(PropUtil.*get*("appid"), PropUtil.*get*("secret"));  **if**(accessToken == **null**) {  **return** "failure";  }  String requestUrl = *scene\_url*.replace("TOKEN", accessToken.getToken());  JSONObject jsonObject = *httpRequest*(requestUrl, "POST", str);  **if** (**null** != jsonObject) {  String tempTickit = jsonObject.toString().split("\":\"")[1];  *QRCode* = "https://mp.weixin.qq.com/cgi-bin/showqrcode?ticket=" + tempTickit.substring(0, tempTickit.length()-6);  }  **return** "failure";  }  **public** **static** String ReadFile(String path){  File file = **new** File(path);  BufferedReader reader = **null**;  String laststr = "";  **try** {  reader = **new** BufferedReader(**new** FileReader(file));  String tempString = **null**;  **while** ((tempString = reader.readLine()) != **null**) {  laststr = laststr+tempString;  }  reader.close();  } **catch** (IOException e) {  e.printStackTrace();  } **finally** {  **if** (reader != **null**) {  **try** {  reader.close();  } **catch** (IOException e1) {  }  }  }  **return** laststr;  }  **public** **static** String get9Num(){  Random r = **new** Random();  **long** l = r.nextInt();  **if**(l<0){  l = -l;  }  **int** total = WeixinUtil.*getNum*(l);  String s = "";  **if**(total == 9){  s = l+"";  }**else** **if**(total >9){  **for**(**int** i=0;i<total-9;i++){  l = l/10;  }  s = l+"";  }**else**{  **for**(**int** i=0;i<9-total;i++){  s += "0";  }  s += l;  }  **return** s;  }    **public** **static** **int** getNum(**long** num){  **int** total = 0 ;  **while**(num!=0){  num = num/10;  total++;  }  **return** total;  }  } |

Figure 45 Code Implement for Wechat Utility

So now the UI of login turns into:

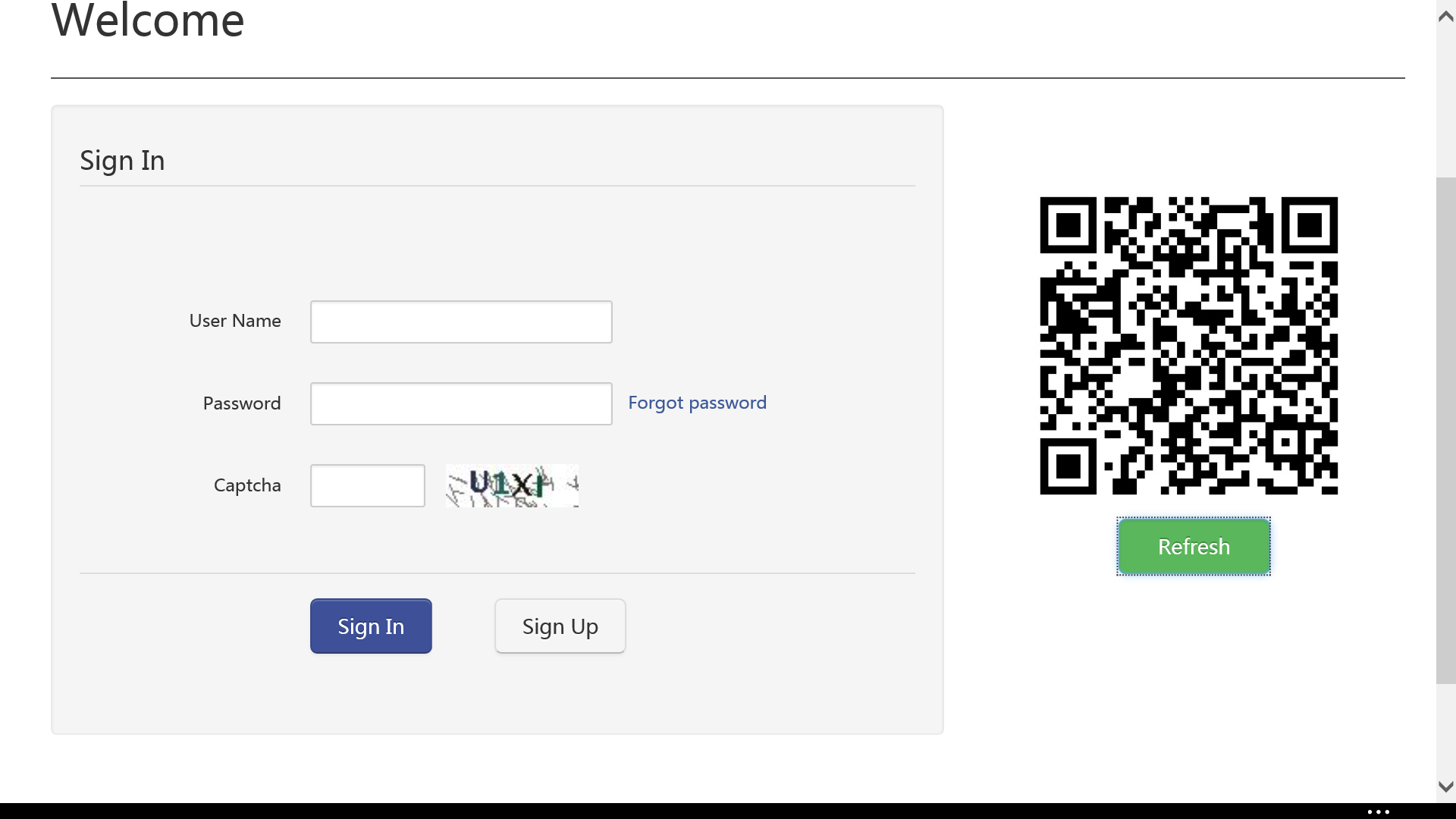


Figure 46 Login with QR

## 7.6 Version 2.1 – Publish to web

First step, purchase space domain.   
Since we use MySQL for database, we choose a UNIX virtual host. 2

Apply ICP record   
According to Ministry of Information Industry requirements, the domestic opening of the site must first apply for ICP website for the record, so after we buy the host successfully, we need first apply for record. Record time is generally about 20 days or so.

步骤阅读Upload website   
During the recording, our domain name is not resolved, or cannot be recognized, the General Chamber of Commerce gives us a temporary registration of a second level domain for accessing, so we can use this domain temporarily for testing our system, while waiting for the results of the recording process.

We use the tool LeapFTP to upload our program, which we'll talk about in detail to upload website.   
Open LeapFTP software.

[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=2)步骤阅读

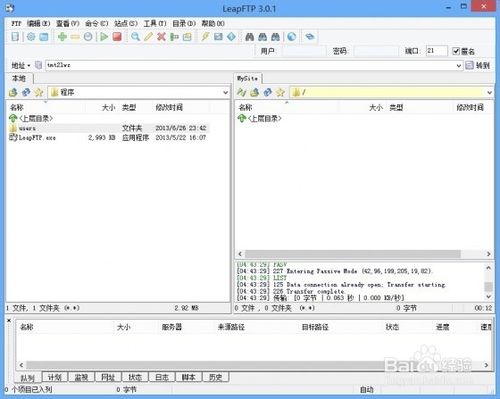
Press F4 to open the "Site Manager." Select the "Site"> "Add"> "Site." Enter the site name.

[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=3)步骤阅读

Site name could be arbitrary, we temporarily name it “MySite”, "FTP host address" is for filling IP address or the domain name, our domain name is 121.40.158.10. Remove the "anonymous login" option, enter "user name", "password" (ftp account password provided by the registrar). The default port is "21", as for the login type, select "Standard." Then click the Apply button.

[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=4)步骤阅读

If the settings are correct, click "Connect" then we can successfully connect the virtual hosts.

[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=5)步骤阅读

1. 7

After the connection, what we can see of LeapFTP can be roughly divided into two parts and the lower part of the left and right. Left area is a local disk, where we can visit the local directory and file. The right is the remote server, where we can manage remote files and local files as well. Right click to create a new directory, double-click to enter the catalog. Area below shows the progress of the file transferring.   
After decompression, the source code’s unzip file is uploaded to the web root directory.

Domain Name   
Domain name resolution and binding now can be done successfully after the recording.   
First login domain management background, where we can make the domain name resolution just by adding a sub-domain of "www" in the A record, and fill in with our host's IP, then click Add. It takes within 2 hours for the domain name to take effect.

Here we take a method to test the domain name to take effect as follows: Start> Run> then enter "cmd", and finally enter "ping www.121.40.158.10.com" command, and it turns out that the domain name come into force.步骤阅读 步骤阅读9

**Domain name bound to space**

We can see that on the virtual hosts registered provider's control panel, there’s a place set to bind space. Only after we bind domain name and space, can we enable domain name to access the contents of the server.   
After finishing these steps, that is buying the domain name space, applying ICP record, uploading website source, DNS, domain binding, now the outside world can access our site. Of course, when we enter the domain name, we can see wp site settings page. After installing and configuring wp, we finally reach the success of publishing our site.

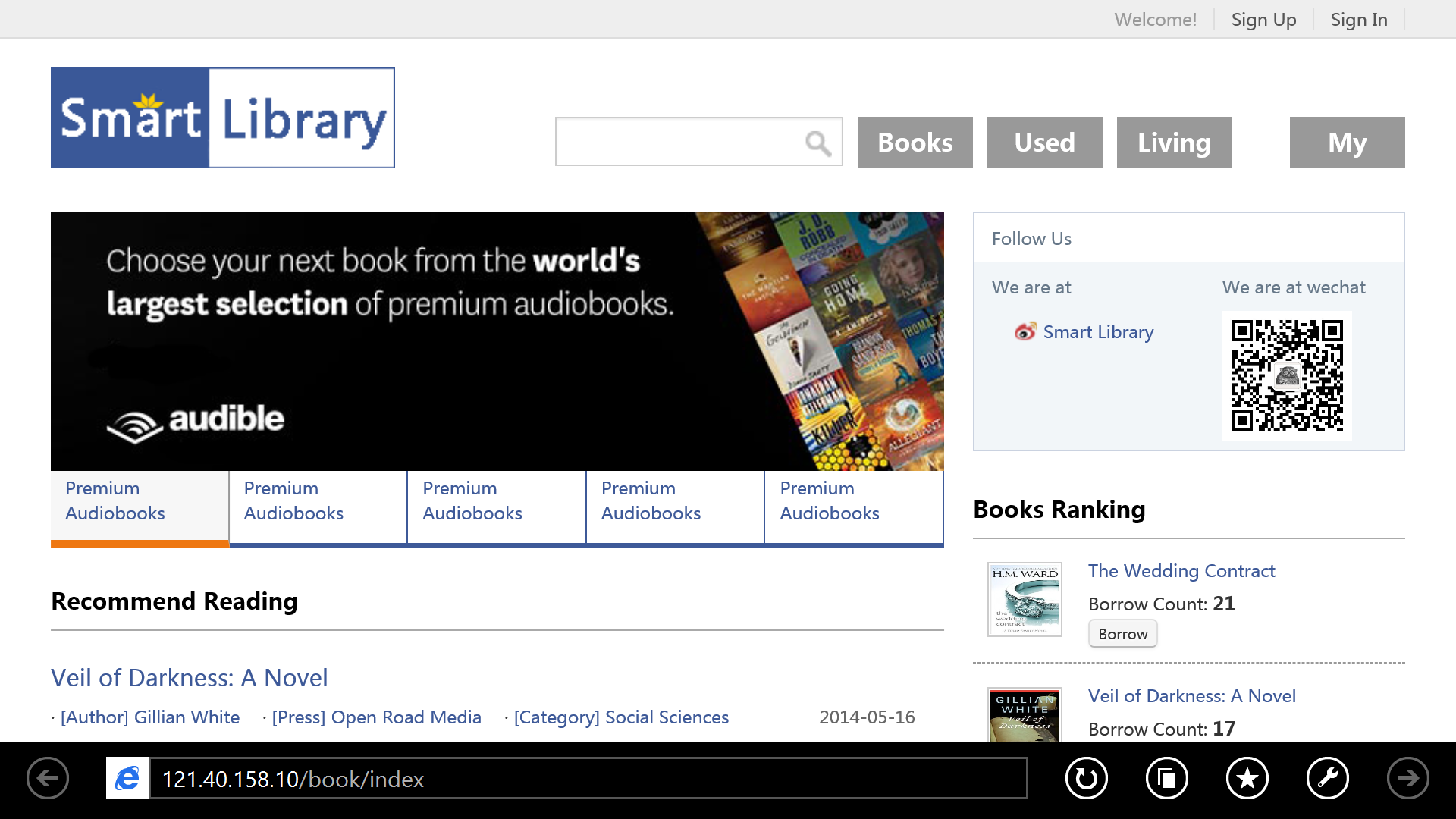


Figure 47 Welcome Page

# 8 Testing

## 8.1 Functional Test

In the process of developing software systems, there are times to face complex issues, therefore, at each stage of the software life cycle, inevitably we produce errors or bugs. So we took different phrase of tests to ensure our system’s quality.

Here’s what we did in testing our library management system. First, all tests should be traced back to user requirements. Then through the view of software engineering, we make the testing fall into 3 levels:

1) Unit testing. Derived from use cases for unit testing.

2) Integration testing and validation testing. This phase is based on the unit test modules, through gradually assembling, testing and correction, it ultimately meets the needs of the software.   
3) Verification test. In this test procedure we found errors of requirements specification.

The unit tested modules together form a subsystem. The test is to inspect coordination and communication between classes inside the module, where the main function of the module was tested.

Then we took integration testing based on unit testing. The tested subsystems are assembled into a complete system for testing, the whole system were tested and debugged after a black-box and a white-box testing of various functional modules. After the integrated testing, the basic functions of the system were tested to be right.

After carrying out the above tests on the entire software system, we took the verification testing, the purpose of verification testing is to verify that the system is indeed able to meet the needs of users.

### Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0001 | Login | Username’ ’  password’111’ | Pops out the dialogue “username or password cannot be empty” | pass |
| 0002 | Login | Username’07050520’  Password’ ’ | Pops out the dialogue “username or password cannot be empty” | pass |
| 0003 | Login | Username’07050520’  Password’112’ | Pops out the dialogue “username or password is not correct” | Pass |
| 0004 | Login | Username’07050520’  Password ‘123’ | Log in successfully | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0005 | Register | Username empty | Pops out the dialogue “username cannot be empty” | pass |
| 0006 | Register | The passwords are not the same when set the passwords | Pops out the dialogue “Passwords not the same” | pass |
| 0007 | Register | Student / Faculty number already registered | Pops out the dialogue “Cannot repeatedly register. Student / Faculty number already registered” | pass |
| 0008 | Register | Correctly fill in the form | Successfully register | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0009 | Borrow | User not logged in | Pops out the dialogue “must log in before borrowing” | pass |
| 0010 | Borrow | The book is already borrowed | The button of borrow should be unable | pass |
| 0011 | Borrow | The user has already borrowed books up to the limited number | Pops out the dialogue “Sorry but you already reach the limit number of borrowing books” | pass |
| 0012 | Borrow | The user is in blacklist | Pops out the dialogue “Can’t borrow because you are in blacklist, if you have any question, contact the librarian please” | pass |
| 0013 | Borrow | The user has overdue book | Pops out the dialogue “Sorry but you have overdue books, please first return the books then borrow new books” | pass |
| 0014 | Borrow | The user state is normal, the book state is available | Successfully borrow the book, user can see the borrow information in his own space | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0015 | Borrow Confirm | A borrowing is successfully made by user | Administrator can see the borrow information in his management page | pass |
| 0016 | Borrow Confirm | Administrator confirms the borrowing | The borrowing take into effect | pass |
| 0017 | Borrow Confirm | Administrator cancels the borrowing because the user doesn’t come to fetch book | The borrowing is canceled and deleted in database | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0018 | Return | Return the book after it’s overdue | Shows overdue days and fined money on administrator’s page | pass |
| 0019 | Return | Return the book in time. | Successfully return | pass |
| 0020 | Return | The book’s ID is not correct | Shows the notification that the book’s ID is not right | pass |
| 0021 | Return | Return a book that is not lend out | Cannot be returned, and shows the message of the book’s state | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0022 | Comment | Empty comment and no scores | Pops out the dialogue “comment and score cannot be empty” | pass |
| 0023 | Comment | Enter text that exceeds the limit of the comment | Shows the message “Please keep the words count in 500” | pass |
| 0024 | Comment | Enter different languages and numbers and signs | The database can support such a combination | pass |
| 0025 | Comment | Enter a normal comment and score | Successfully comment | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0026 | Search book | Book ID not exist | Shows the message “Book ID does not exist.” | pass |
| 0027 | Search book | Book name in different languages | The database can support and gives out the result of searching | pass |
| 0028 | Search living book | Living book’s ID not exist | Shows the message “Living Book ID does not exist.” | pass |
| 0029 | Search living book | An existed living book name | Shows the living book with a profile | pass |
| 0030 | Search user | User Id not correct | Shows the message ”user id not correct” | pass |
| 0031 | Search user | User not exist | Shows the message ”user id not exist” | Pass |
| 0032 | Search user | User Id existed and correct | Shows the information of the user | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0033 | Add a book | add a book with an ID already existd | Pops out the dialogue “Book Id cannot be repeated” | pass |
| 0034 | Add a book | Some information is empty | Pops out the dialogue “Please complete the information of the book” | pass |
| 0035 | Add a book | A unique ID and completed information | Successfully add a book | pass |
| 0036 | Delete a book | Delete a book that is borrowed or renewed | Shows the message “cannot delete because the book’s borrowed” | pass |
| 0037 | Delete a book | Delete a book that is available | Successfully delete the book | Pass |
| 0038 | Modify a book | Modify the ID of the book with an existed ID | Pops out the dialogue “Book Id cannot be repeated” | Pass |
| 0039 | Modify a book | Modify the book’s state | Gives the warning and change the state | Pass |
| 0040 | Modify a book | Modify the book’s ISBN | Gives the warning and change the ISBN | Pass |
| 0041 | Modify a book | Modify the book’s title, author, price, press, publishing date | Gives the warning and change the state | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0042 | Add a living book | Add a living book with existed ID | Pops out the dialogue “Living book Id should be unique” | pass |
| 0043 | Add a living book | Add a living book with uncompleted information | Pops out the dialogue “Please complete the information of the living book” | pass |
| 0044 | Add a living book | Add a living book with unique ID and completed information | Successfully add the living book | pass |
| 0045 | Delete a living book | Delete a living book | Gives the warning and delete successfully | pass |
| 0046 | Pull the living book into blacklist | Pull the living book into blacklist | Successfully pull. The living book cannot be searched by user | Pass |
| 0047 | Modify a living book | Modify a living book’s name, timetable, profile, category | Successfully modify | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0048 | Register admin | Admin name empty | Pops out the dialogue “Admin name cannot be empty” | pass |
| 0049 | Register  Admin | Admin passwords not the same | Pops out the dialogue “Passwords not the same” | pass |
| 0050 | Register  Admin | Other must fill-in information empty | Pops out the dialogue “Please fill in the blank of must information” | pass |
| 0051 | Register  Admin | Correctly fill in the form | Successfully register | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0052 | Modify user password | Username empty | Pops out the dialogue “username cannot be empty” | pass |
| 0053 | Modify user password | The passwords are not the same when set the new passwords | Pops out the dialogue “Passwords not the same” | pass |
| 0054 | Modify user password | The password is empty | Pops out the dialogue “Please set the password” | pass |
| 0055 | Modify user password | Correctly fill in the form | Successfully modify the user password | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0056 | Post a used book for sale | Book name empty | Pops out the dialogue “book name cannot be empty” | pass |
| 0057 | Post a used book for sale | Not completed information | Pops out the dialogue “Please complete the information” | pass |
| 0058 | Post a used book for sale | Introduction exceeds the limit words count | Pops out the dialogue “Please make the introduction less than 500” | pass |
| 0059 | Post a used book for sale | Correctly fill in the form | Successfully post a used book for sale | pass |
| 0060 | Post a used book for purchase | Book name empty | Pops out the dialogue “book name cannot be empty” | pass |
| 0061 | Post a used book for purchase | Not completed information | Pops out the dialogue “Please complete the information” | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Description | Expect | Result |
| 0062 | Delete a post | Delete a post with admin privilege | Gives the warning and successfully delete | pass |
| 0063 | Delete many posts at a time | Delete a list of posts at a time with admin privilege | Gives the warning and successfully delete | pass |
| 0064 | Add a post then delete | Add a post with user status and then delete with admin privilege | Gives the warning and successfully delete | pass |
| 0065 | Add a list of posts and then delete some | Add a list of posts with user account and then delete some of them at a time using admin account | Gives the warning and successfully delete | pass |

**Data Consolidation**   
Due to small size of the system, the input and output data type is simple, so we use the test data manually processed to facilitate evaluation of the appropriate form.

We took the testing and make sure that the system complete functional requirements correctly.  
As for the efficiency or usability of the system, we tried to take some usability testing covering a part of the main functions of the system, which is illustrated in next part.

## 8.2 Usability Test

**Library Management System test description**

We use an approach of usability testing that involves testing a relatively small, representative group of users in order to produce reliable results, reduce testing costs, and reduce the amount of time spent conducting and analyzing test results. Industry studies confirm that carefully designed tests with a small number of test participants (4-6, typically) discover at least 80% of the usability problems revealed in formal tests with much larger numbers of test participants.

**Test format**

The complete test for each test participant involved the following activities:

Pre-test activities consisted of phone calls to qualify the test participants and completion of a pre-test questionnaire designed to confirm that the test participant met our user profile requirements.

Formal testing in a usability lab involved testing individual participants in a lab environment set up much like a dorm, classroom or library. Users completed real-life scenarios using Library Management System while we recorded their activities on video and observed them through a one-way mirror. The usability test room (where test participants worked) was furnished with three video cameras set up to tape test participants from various angles.

Walkthrough—To test our scenarios and the general “do-ability” of the test, we invited a sample test participant to complete the test scenarios in the lab while we logged the results. We used the results from our walkthrough test participant to confirm that the test scenarios were readily understandable and to review our timing assumptions. The walkthrough experience gave us a chance to revisit some awkward wording and revise our test time limits.

Pilot test—After refining the test scenarios and questionnaires, we conducted a pilot test under authentic test conditions to make sure our revisions had corrected all critical problems with the test.

Post-task and post-test activities consisted of short questionnaires administered between scenarios and a longer questionnaire and interview administered after the final scenario. The test team used the final interview as an opportunity to ask test participants about specific issues that arose during their individual tests.

Summary sessions conducted after each test allowed the evaluation team to quickly examine test results and categorize usability issues discovered in the test.

**Test times**

We tested Library Management System both on weekday and on weekends. We tested two users per 1-hour session.

**Test participant selections**

We decided to test two groups of users, novice users and advanced users. By testing novices, we were more likely to collect information about the intuitiveness of the application. By targeting advanced users, we could be confident of assembling a group of people capable of performing the more difficult tasks we were asked to test.

Novice Profile: We selected novice test participants who met our requirements of having basic computer experience, including keyboarding skills, Windows experience, and the ability to access and navigate a Web browser.

Advanced Profile: We selected advanced users who met the minimal requirements of the novice users, but who in addition had previous experience using other library management programs (except for this Library Management System). We limited this study to participants who had extensive Web experience (6 months +).

**Test problems**

Test participant selection problems

We originally recruited and qualified three novice test participants and three advanced participants, each group containing a mixture of ages and genders. However, during one evening testing session, Library Management System was unexpectedly taken down for maintenance. We had to excuse our sole male novice participant test participant and recruit another on short notice. The replacement participant subject was a 45+-year-old female with advanced user skills.

Because of this substitution, our final test sample was composed of four advanced participant users and only two novice users.

Wide variations in Library Management System response

We noticed a considerable difference in system responsiveness depending on the time of day that the tests took place. Library Management System was very slow on weekday evenings, but quite responsive on weekend mornings. This is important to note because it may help explain why there is such a wide divergence of opinion among the test participants concerning the "speediness" of Library Management System.

**Scenario and questionnaire design**

We created a set of scenarios designed to test the issues discussed previously. Each scenario was designed to require between 15-25 minutes to complete. We constructed the scenarios to be as real-world as possible, containing multiple related tasks (e.g. searching a book and then borrowing it).

The questionnaires administered after each scenario and after the complete tests were designed to collect some of the "qualitative" data omitted from the formal testing. The questionnaires quizzed test participants about their perceptions about Library Management System and how well they understood and liked the product.

**Scenario design**

We revised our test scenarios slightly because of some problems that occurred during testing. The final scenarios and tasks were as follows:

|  |
| --- |
| Scenario 1: Registering for Library Management System and Logging On (time limit: 10 minutes) |
| Scenario 2: Searching, sorting and borrowing books (time limit: 15 minutes)  Task A: Search a book by the title  Task B: Sort the results by the author  Task C: Click one book and borrow it |
| Advanced user scenario  Scenario 3: Search a book by the title, ISBN, author, press, published date (time limit: 15 minutes)  Task A (Discontinued because application wouldn't support a search for non-existing Library Management System books).  Task B: Sort the results by the title, author, press, published date  Task C: Borrow a book  Task D: Check the books’ comments |
| Scenario 4: Cancel borrowing and Leaving Library Management System (time limit: 10 minutes)  Task A: cancel the previous borrowing  Task B: Sign off from Library Management System. |

**Time limits**

We allotted 50 minutes for the test participants to complete all four scenarios.

When we originally designed the scenarios, we believed that because of their limited abilities and experience, novice test participants might not have enough time to attempt Scenario 3, which tests the advanced Library Management System tasks.

However, because our novice users were able to complete scenarios 1, 2, and 4 reasonably quickly (setting aside system-imposed delays), we decided to allow them to attempt the more advanced tasks. As a result, the test scenarios and time allotments are the same for the two groups of test participants.

**Test questionnaires and interview design**

We designed a series of questionnaires to qualify potential test participants and to collect subjective feedback from the test participants during and after testing. The Appendix contains the actual questionnaires.

Pre-test questionnaire

To qualify test participants, we asked a series of questions designed to collect demographic information and to assess their level of computer and Internet experience. We accepted for testing those test participants who met our minimum requirements for novice or advanced users.

Post-task questionnaires

After the test participants completed each scenario, we administered a post-task questionnaire. The questionnaire was designed to capture feedback about Library Management System tasks while the test participant's memory was fresh. The questionnaires asked test participants to rate the ease or difficulty of the tasks, describe whether Library Management System's terminology relative to the task was clear, and provide free-form comments concerning the tasks.

Post-test questionnaire

After the test participants completed the final scenario, we administered a questionnaire designed to capture their general opinions about Library Management System. The questionnaire asked test participants to rate their interest in having online library management, their impressions about Library Management System's speed and responsiveness, and their overall satisfaction with the product.

Post-test interview

After each of the test participants completed the post-test questionnaire, the test briefer (the evaluation team's liaison with the test participant) scanned the questionnaire and asked the test participant to explain some of the responses. As the need arose, the briefer occasionally presented the test participant with additional questions that the evaluation team compiled while observing the test.

**Usability criteria**

To categorize the test observations, we applied a usability test analysis model that categorizes product usability problems according to whether they fail to meet any of the following 102 criteria:

1. Concept: Does the product use effective metaphors? Is it intuitive?

2. Consistency: Does the product look and perform similarly through all parts of the application?

3. Content: Is the content accurate, appropriately complex, and provided in the right amount? 4. Feedback: Does the product provide appropriate feedback to the user?

5. Interaction Model: Are user responses and other system interactions handled according to established models?

6. Navigation: Is it easy to get where you want to go in the product?

7. Terminology: Is the interface's language easy to understand for the audience(s)?

8. User Assistance: Does the product supply an appropriate amount of user help (e.g., online help, how to get customer service, other instructions)?

9. User Preference: Does use of the product cause difficulty for individual users in other ways not covered in these categories (e.g., does it favor "mouse users" over "keystroke users"; is it perceived to be too slow)?

10. Quality Assurance: Is the product robust? Does it fail in ways that prevent users from performing tasks?

Our test results produced findings in most of these areas, which we rated using the rating scales discussed in the following section.

**Scope and severity ratings**

After identifying usability problems, we rated their scope and severity. Scope refers to how widespread the problem was throughout the product, and severity codes rate the seriousness of the problem. Scope Local problems consisted of problems that occurred only in a particular part restricted to a particular part of the application, while global problems indicated far-reaching design flaws that occurred with consistency throughout the product. In general, global problems tend to be critical to correct, because they affect usability of the entire application, have far-ranging effects, and tend to be critical to correct. However, some local problems are critical enough to hamper severely the users' ability to perform key tasks.

Severity

We used the following severity codes to classify the seriousness of the problems we discovered:

1 - Prevents completion of a task

2 - Causes significant delays in completing a task

3 - Causes minor usability problems, but users can complete the task

4 - Minor annoyance - does not significantly impact usability, but should be corrected if time allows

# 9 Conclusion

## 9.1 Technical Complexity

One of the technical complexities we encountered in the system is how to do the recommendation, there are lots of algorithms for recommendation, typically producing a list of recommendations in one of two ways - through collaborative or content-based filtering. [Collaborative filtering](http://en.wikipedia.org/wiki/Collaborative_filtering) approaches build a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users; then use that model to predict items (or ratings for items) that the user may have an interest in. [Content-based filtering](http://en.wikipedia.org/w/index.php?title=Content-based_filtering&action=edit&redlink=1) approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties.

We did a study for the algorithms and finally decided to make a simple and straight algorithm out of the principle which indicates that if the book is read by these users, and these users mostly read that book, then we recommend that book to the current user. What’s more, we use tags to find the books that are of the same kind, so if the book has the tags, we also look up other books that has the same tags and are rating high scores to recommend to the user.

## 9.2 Innovation

Living library is a new concept to apply to the school’s library, we have illustrated the concept in the previous chapter, however it’s easy to develop and hard to apply, the key solution is to provide a set of reasonable and applicable rules to make the system really work. After several talks with the leaders and administrators of the library of JSNU, we put forward such a set of rules to apply:

**Select living books**

1. Form a committee for living library to be responsible for the selecting living books
2. The committee is made of 8 people that are professors from several departments and advanced administrators of the library
3. There are 2 steps for contestants to be selected for living book. First, submit the application form online, second, take the interview of the committee members if pass the trial.
4. To pass the trial, there should be at least 4 members out of 8 to admit, to pass the review, the contestant need more than 5 members’ admissions.

**Basic requirements and steps to apply for living book**

1. Must be student or faculty member of the school
2. Must have some contributions at a certain area, e.g. published paper or medals
3. Must have good moral and strong psychological quality
4. In order for the usability and vitality of the living library, the applicant must provide at least 2 periods of available time in a week, and each period must last 30 minutes.
5. The living library is a nonprofit public service, and all the living books are volunteers to make contribute to living library, the school will only give the living books compensatory payment
6. The applicants can download the application form from the library management system’s home page and submit the form to the committee’s mailbox
7. The committee will notify the applicants the result of the trial by email or text message in 14 working days. Those who pass the trial will be told the time and place of the interview.
8. After the interview, the committee will notify the interviewees the result by email or text message in 3 working days. Those who pass the interview will be admitted to living library.

**Steps of quitting the living library**

1. The applicants can download the form on the library system’s home page, fill in the form and then submit it to the committee’s mailbox.
2. The committee member will verify the form and notify the applicant the result by email or text message.
3. Once quit the living library, the applicant cannot apply again in 3 months, so think carefully before quitting.

**Management for living library**

1. If the living book have more than 5 times of breaking the promise with the user, then administrator will give warning, if it is more than 10 times, then administrator will disqualify the living book.
2. If the user have more than 5 times of breaking the promise with the living book, then administrator will give warning, if it is more than 10 times, then administrator will add the user to blacklist.
3. If users or the living books say or do something related with discrimination, attack, trick, or sexual offend, once verified by administrator, the user or the living book will be added into blacklist or disqualified.
4. If the comments have content related with discrimination, attack, trick, or sexual offend, the user or the living book who write the comment will be warned by administrator the first time, if it is the second time, the comment maker will be added to blacklist at once.

## 9.3 Future Enhancements

There’s one kind of resource of the library that we don’t include in the current system, that is E-books, since we focus on the basic management of the library and the innovated living library, well the E-books of the library cannot be ignored easily, so in next version, we will put the e-books management into our system.

We will also consider to provide with the Douban website connecting service, you can see the evaluation of the books from readers on Douban. If the book can’t be borrowed or is not the collections of the library, you can also directly connect to a number of online bookstores, where you can purchase the book online.   
Simple and practical journal directory pushing service. Want to know the professional journal Table of Contents? Well, we will let you customize autonomy and regularly push the latest directory to you.   
Recommended the purchase. We often see some great books on Douban, but the library does not have the books, how to do? In next version, you can immediately apply to the library with the purchase recommending book, so that administrators will consider to buy the books you recommend.

What’s more, we’d like to provide the best experience in searching with:

1. **High-quality academic search engine**
2. **Give the search results relevance ranking**
3. **full-text search capabilities**
4. **sophisticated advanced search functions**

# 10 Appendices

## 10.1 Reference Guide

### 10.1.1 Books

1. Wang Lifu. Software Engineering (Second Edition), Beijing: Peking University Press .2009

2. Horstmann Gay S.).Gary Cornell.Java core technologies: Basics (eighth edition) Machinery Industry Press, 2008.

3. Yu Yan, Zheng Archie.J2EE Application of Electronic Industry. Publishing House.2009 Tutorial: 40 to 45.   
4. Cho Jin. MyEclipse development techniques in Detail. Chemical Industry Press, 2010.   
5. Wu Bo, Rui Chi Nan, Zhang Li Juan. Java Advanced Programming practical tutorial. Tsinghua University Press, 2008: 43 to 89.   
6. Cai Jianping Software Development Case Comprehensive Tutorial: Linux, GCC, MySQL, Socket, Gtk + and open source case. Tsinghua University Press (2008).   
7. Klein Berg (Eric Clayberg), Roubaix (Dan Rubel), Pei. Eclipse plug-in development (the original book version 3) Machinery Industry Press, 2010.   
8. Chen Guilin. Database program design (2nd edition).China Water Power Press, 2010: 90 to 98.   
9. Zhao Bin. Software testing techniques classic tutorial (2nd edition). Science Press, 2011: 10.   
10. Chen Tianhe. Eclipse, Struts, Hibernate, Spring Integrated Development Collection. Electronic Industry Press, 2008

### 10.1.2 Websites

One example of living library: <http://living.lib.sjtu.edu.cn/>

Struts framework: <http://struts.apache.org/>

Spring framework: <http://projects.spring.io/spring-framework/>

Hibernate framework: <http://hibernate.org/>

MyEclipse IDE: <http://www.myeclipseide.cn/>

ExtJS: <http://www.sencha.com/>

WeChat: <http://www.wechat.com/zh_TW/>

### 10.1.3 Our online library management system address

http://121.40.158.10/

## 10**.**2 Usability test questionnaire

Library Management System Usability Test Final Report

Pre-Test Questionnaire

|  |
| --- |
| **Thank you for considering being a volunteer for our Usability Test**. We will work to make sure the test environment is pleasant and fun for you, with a casual dinner being served.  The results from our Usability Test will be used to help improve a computer software product’s ease of use.  **Please answer the following questions.** We will use your answers to determine if you will be a participant in our Usability Test.  **The testing will take place on July 22 and July 29 from 6-8 p.m.** The usability test will require 1 1/2 hours of your time.  Please place a check next to the dates on which you are available, if any.  July 22 \_\_\_\_\_\_ July 29 \_\_\_\_\_\_ Either Date \_\_\_\_\_\_ Neither Date \_\_\_\_\_  **Please disregard the rest of the questionnaire if you are not able to attend one of the dates**. **Thank you again for your consideration.** |

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Home Phone: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Work Phone: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Age: [ ] 15-20 [ ] 21-30 [ ] 31-40 [ ] 41-50 [ ] 51 or above

Sex: [ ] Male [ ] Female

[ ] Right handed [ ] Left handed

Please answer the following questions about your computer experience:

1. Do you use a personal computer?

[ ] Yes [ ] No

If you answered “no,” please disregard the remaining parts of the questionnaire.

2. What kind(s) of programs have you worked with? Check all that apply.

[ ] Word Processing [ ] Spreadsheets [ ] Graphics [ ] Other(s) specify\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. How long have you been using personal computers?

[ ] 0-3 months [ ] 4-6 months [ ] 7-9 months [ ] 10-12 months [ ] More than 12 months

4. Have you ever used a Web browser? [ ] Yes [ ] No

If you answered “no,” please proceed to question 7.

5. Which Web browser have you used? Check all that apply.

[ ] Microsoft Internet Explorer [ ] Netscape Navigator [ ] Other(s) specify\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. How long have you been browsing the Web?

[ ] 0-3 months [ ] 4-6 months [ ] 7-9 months [ ] 10-12 months [ ] More than 12 months

7. Which Internet Service Provider(s) do you use? Check all that apply.

[ ] AOL [ ] Prodigy [ ] CompuServe [ ] Mindspring [ ] other (specify)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[ ] I do not use/have an Internet Service Provider

8. How long have you been using your work/school location’s network?

[ ] 0-3 months [ ] 4-6 months [ ] 7-9 months [ ] 10-12 months [ ] More than 12 months

9. Do you use the library system for borrowing a book?

[ ] Yes [ ] No

If answered “no,” please proceed to question 12.

10. Which function of the system do you use? Check all that apply.

[ ] Search a book [ ] Borrow a book

[ ] browse new books [ ] other (specify)

11. How long have you been using your school’s library system?

[ ] 0-3 months [ ] 4-6 months [ ] 7-9 months [ ] 10-12 months [ ] More than 12 months

12. What do you borrow books for? Check all that apply.

[ ] Study [ ] Kill time [ ] Other (specify)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

13. Do you know how to search a book on the internet? [ ] Yes [ ] No

17. Do you know how to make the keywords out of a piece of information? [ ] Yes [ ] No

18. Do you know how to create and send a simple email message? [ ] Yes [ ] No

Thank you for completing our questionnaire. We greatly appreciate your consideration and time. We will be in contact with you if you are selected to participate in our Usability Test. Thank you!

**Post-test questionnaire**

We reviewed the results with the test participants in post-test interviews.

|  |
| --- |
| **Thanks for completing the usability test.**  Please answer the following questions about your experience with Library Management System. We will use your answers to provide important feedback to Library Management System's marketing and development staff. |

1. On the following scale, rate your need for / interest in having an online library management system:

No interest/need [ ] Low interest/need [ ] Don't feel strongly either way [ ]

Moderate interest/need [ ] High interest/need [ ]

2. On the following scale, rate your impression of Library Management System's speed and responsiveness:

Very slow [ ] Moderately slow [ ] Neither fast nor slow [ ]

Moderately fast [ ] Very fast [ ]

3. Will you use Library Management System in the future?

Never [ ] Rarely [ ] Sometimes [ ] Fairly frequently [ ] Very frequently [ ]

4. On the following scale, rate how highly you would recommend Library Management System to your friends and associates:

Would NOT [ ] Would recommend [ ] Don't feel strongly either way [ ] Would probably recommend [ ] Would strongly recommend [ ]

5. If you plan to use Library Management System at all in the future, please indicate how you might use it (Check all that apply):

Weekends for killing time [ ] Weekends for studying [ ] Weekdays for homework [ ] At dorm [ ] At classroom [ ] Not applicable--will not use [ ]

6. What did you like MOST about Library Management System?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8. What did you like LEAST about Library Management System?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9. What would you change about Library Management System?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10. Optional: Please add any other comments about Library Management System that might be useful in helping Library Management System staff improve this product:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 10.3 Application form of living book

**Apply to Join:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name |  | Age |  |
| Major |  | Student/Faculty Number |  |
| Phone |  | Email Address |  |
| Skype Account |  | Address |  |
| Provide at least 2 periods of available time (at least 30 minutes per period) |  | | |
| Profile |  | | |
| Specific Area and the relating expertise |  | | |
| Achievements |  | | |

**Apply to Quit:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name |  | Age |  |
| Major |  | Student/Faculty Number |  |
| Phone |  | Email Address |  |
| Skype Account |  | Address |  |
| The reasons for quitting living library |  | | |