**书趣详细设计说明书**

[1 目的 2](#_Toc7832)

[2 代码框架描述 2](#_Toc7832)

[2.1 目录结构 2](#_Toc5445)

[2.2 模块配置文件说明 2](#_Toc24659)

[3 子模块详细设计说明 2](#_Toc2843)

[3.1 登录和注册模块、兴趣选择模块 2](#_Toc23389)

[3.2 书籍检索模块 3](#_Toc23420)

[3.4 我的主页 16](#_Toc8287)

[3.5 管理书库 16](#_Toc31267)

[3.6 管理社区 17](#_Toc19194)

**术语说明：**

## **动态action：前端和后台交互方式之一，前端通过超链接的方式可将参数传递给后台，当用户点击超链接时即进行参数的传递，并运行后台程序。运行后会跳转页面。**

## **form表单传递：前端和后台交互方式之一，前端通过在form标签中指定action的值作为传递路径，method的值作为传递方式，将所有input的值传递入后台，input的属性name值和action的变量一一对应。触发submit事件后即进行传递，并运行后台程序。运行后会跳转页面。**

## **Ajax传递：前端和后台交互方式之一，前端通过Ajax事件，将参数以json的形式传递给后台，并指定传递方式，异步还是同步，后台程序执行后可返回参数，触发Ajax中的返回函数。运行后不进行页面跳转。**

# **1 目的**

本说明旨在说明项目各部分实现思路及所涉及的数据结构等，方便开发人员对项目架构、设计有着更深入的理解，并为编码提供开发思路。

**2 代码框架描述**

系统源代码按模块开发，同一模块文件存放在同一目录下，各模块目录统一存放在根目录下。各模块视图层涉及的CSS、JS、images等文件均存放在相应目录下的对应文件夹中。根目录下只存放首页文件，首页设计的CSS等存放在根目录下的对应文件夹下。后台整体采用SSH框架开发，MVC设计模式。

## 2.1 目录结构

系统源代码文件存放的目录、子目录，及其包含的文件列表。如下：

详见附件5

## 2.2 模块配置文件说明

配置文件主要包括SSH框架的struts.xml、application.xml、xxx.hbm.xml主要处理各模块之间的逻辑关系，以及VO和数据表之间的对应关系。

# **3 子模块详细设计说明**

本部分主要对各模块实现原理做详细说明

## 3.1 登录和注册模块、兴趣选择模块

**3.1.1 登录**

1. 验证码的生成由Java的awt包生成，前端通过img标签src属性调用，验证码内容存储在session中

2. 验证码的验证通过Ajax调用验证程序的JSP文件，并传递验证码输入框内容，JSP文件取得验证码输入内容，并将存储在session中的验证码内容取出进行匹配验证

3. 登录操作通过form表单将各接口传入后台验证，验证程序使用HQL进行数据库内容匹配，并将用户对象存储在session中。权限初始为1，权威初始为0

4. 记住密码被选中后，后台通过Cookie进行用户名和密码的存储，前台可取出Cookie进行输入框的填充

**3.1.2 注册**

1. 将注册信息输入完毕后点击注册按钮，弹出对话框选择兴趣
2. 确认兴趣后将数据传入后台进行注册操作，未选择则不进入
3. 用户名的动态验证通过Ajax进行后台传递输入的用户名，传输给用户信息的action，验证是否在数据库中存在，返回存在与否的信号

## 3.2 书籍检索模块

**3.2.1 在本库中检索**

1. 进入检索页面时进行JS判断，调用后台程序进行数据库内容的分页输出。其中图片为将图片存储路径传递给前台img的src。若数据库中没有该图片路径，则输出预制图片路径
2. 点击分类时，将该分类的名称用动态action传至后台，后台在索引库中进行检索，并将取出的数据分页传给前台
3. 高级检索部分使用form表单将搜索信息传输至后台，后台在索引库中进行检索，并将搜索结果分页传给前台
4. 普通检索将关键词传至后台，后台在索引库中进行检索，将查找结果分页传给前台
5. 点击收藏按钮，Ajax将书籍ID传给后台，后台接收ID号并用HQL执行删除操作，返回成功与否信号
6. 点击评分后，Ajax将书籍ID号及分数传至后台，后台借助评分模型该ID对应书籍的最终分数并存入数据库，再计算并修改热门度，之后再将用户对象和分数存储至评分表中，返回成功与否信号
7. 选择排序方式后，超链接动态action将排序方式编号传至后台，（每种排序方式对应一种编号，按需求分析顺序由1依次往下排），后台在索引库中检索并排序，将结果分页传回前台
8. 商铺竞价：书籍详细页加载完毕后，使用Ajax将当前书籍ID传到后台，后台根据ID取出对应书籍对象，获取书名，使用爬虫爬取京东对应第一个商品的价格、链接，以json格式传回前台，前台对json进行解析后显示。中间过程前台始终在对应位置显示加载动画。
9. 国图推荐：国图检索时将检索结果存入数据库并进行聚类。进入书籍详细页时待页面加载完毕，使用Ajax将当前书籍ID传到后台，后台根据ID取出对应书籍对象，获取书籍性能度量书籍并加入聚类模型找到同一类的聚类簇。取该聚类簇样本序号，并在国图检索存储数据库中找到对应对象，以json格式传回前台，前台对json进行解析后显示，中间过程前台始终在对应位置显示加载动画。

**3.2.1.1 评分模型算法**

输入：评分分数 = bookScore;

书籍ID = bid;

用户对象 = user.

过程：函数scoreBook(bookScore, bid, user)

1: 根据user获取权威值userAuth;

2: if userAuth < 0 then

3: return

4: end if

5: 根据bid获取书籍对象book;

6: 根据book获取评分总分数score，评分总人数people;

7: 计算最终分数newScore = (score \* people + bookScore \* userAuth \* 0.1) / (people + 1);

8: people = people + 1;

9: 存储newScore和people;

输出：评分操作成功与否

注：本模型评分总分数范围为0-10，最大值情况为 5 \* 20 \* 0.1 \* 评分总人数 / 评分总人数（即所有评价用户权威均为最大，评分均为最大）

**3.2.1.2 猜你喜欢算法**

本算法系改进版基于相似物品的协同过滤算法，核心公式为余弦相似度计算公式

输入：书籍ID = bid.

过程：函数recommend(bid)

1: 获取全部书籍集合list

2: 根据bid获取本物品书籍对象book

3: list - book

4: 获取list大小size

5: sum = 0

6: for i = 0,1,2,...,size do

7: 计算本物品书籍book和list(i)的相似度calCorrelation(i);

8: if calCorrelation(i) = -1 then

9: continue;

10: end if

11: if calCorrelation(i) > 0.7 then

12: list(i)加入推荐书籍集合bookList中;

13: sum = sum + 1;

14: end if

15: if sum = 4 then

16: break;

17: end if

18: end for

19: if bookList = 空 then

20: 根据book获取书籍类别bookInte;

21: 获取4个同类别书籍对象加入bookList;

22: end if

输出：推荐书籍集合bookList

**3.2.1.2.1 相似度算法**

输入：本物品书籍名称currentBookName;

待测物品书籍名称testBookName.

过程：函数calCorrelation(currentBookName, testBookName)

1: sum1 = 0;

2: sum2 = 0;

3: sum3 = 0;

4: 根据currentBookName和testBookName获取十个同时对本物品和待测物品评分的用户集合userList

5: if userList = null then

6: return -1;

7: end if

8: if userList = 空 then

9: return -1;

10: end if

11: 获取userList的大小size

12: for i = 0,1,2,...,size do

13: 根据currentBookName和userList(i)获取本物品书籍评分分数scoreI;

14: 根据testBookName和userList(i)获取待测物品书籍评分分数scoreJ;

15: 根据currentBookName和userList(i)计算本物品书籍残差residualI;

16: 根据testBookName和userList(i)计算待测物品书籍残差residualJ;

17: sum1 += (scoreI - residualI) \* (scoreJ - residualJ);

18: sum2 += (scoreI - residualI) \* (scoreI - residualI);

19: sum3 += (scoreJ - residualJ) \* (scoreJ - residualJ);

20: if sum2 = 0 || sum3 = 0 then

21: return -1;

22: else

23: return sum1 / Math.sqrt(sum2 \* sum3);

24: end if

25: end for

输出：本物品和待测物品书籍相似度

**3.2.1.2.2 获取同时对本物品和待测物品评分用户算法**

输入：本物品书籍名称currentBookName;

待测物品书籍名称testBookName.

过程：函数getScoreUser(currentBookName, testBookName)

1: num = 0;

2: 根据currentBookName获取对本物品评分的对象集合scoreList

3: if scoreList = 空 then

4: return null;

5: end if

6: 获取scoreList大小size

7: for i = 0,1,2,...,size do

8: 根据testBookName和scoreList(i)的用户名scoreUser(i)获取同时对待测物品书籍评分的对象集合seScoreList;

9: if seScoreList != 空 then

10: 将scoreUser(i)加入同时对本物品和待测物品评分用户集合list中;

11: num = num + 1;

12: end if

13: if num = 10 then

14: break;

15: end if

16: end for

输出：同时对本物品和待测物品评分用户集合list

**3.2.1.2.3 书籍残差算法**

输入：指定用户用户名userName;

书籍名称bookName.

过程：函数calResidual(currentBookName, testBookName)

1: sum = 0;

2: 获取全部评分对象集合list;

3: 获取list大小size;

4: for i = 0,1,2,...,size do

5: sum += list(i)的分数scoreBook(i);

6: end for

7: 计算总体平均评分u = sum / size;

8: sum = 0;

9: 根据bookName获取该用户所有评分书籍分数集合list;

10: 获取list大小size

11: for i = 0,1,2,...,size do

12: sum += list(i)的分数scoreBook(i) - u;

13: end for

14: 计算书籍偏置calBookBias = sum / (25 + size);

15: sum = 0;

16: 根据userName获取用户所有评分书籍分数集合list;

17: 获取list大小size;

18: for i = 0,1,2,...,size do

19: sum += list(i)的分数scoreBook(i) - u - list(i)的书籍偏置(同8-14);

20: end for

21: 计算用户偏置calUserBias = sum / (10 + size);

22: 计算残差residual = u + calBookBias + calUserBias;

输出：残差residual

**3.2.1.2.4 聚类算法**

输入：国图检索样本集sampleList = {x1,x2,...,xm}.

过程：

1: 获取聚类簇集合clusterList = {a1,a2,...,am};

2: if clusterList = null do

3: 随机取5个样本作为初始均值向量{u1,u2,...,uk};

4: repeat

5: 令Ci = null;(1<=i<=k)

6: for j=1,2,...,m do

7: 计算样本xj与各均值向量ui(1<=i<=k)的距离：dji = ||xj - ui||2;

8: 根据距离最近的均值向量将对应xj划入该簇

9: end for

10: for i=1,2,...,k do

11: 计算新均值向量ui’;

12: if ui’ != ui then

13: 将当前均值向量ui更新为ui’

14: else

15: 保持当前均值向量不变

16: end if

17: end for

18: until 当前均值向量均未更新

19: else

20: for i=1,2,...,m do

21: 计算当前样本和各簇均值向量距离dji;

22: 根据距离最近将样本存入对应簇并更新均值向量

23: end for

24: end if

**3.2.2 在国家图书馆检索**

1. 将输入框中内容传入后台，使用HtmlUnit爬取国家图书馆页面，使用Jsoup进行页面分析，爬取需要的内容
2. 对于国家图书馆内容分页的实现，利用HtmlUnit对JavaScript的模拟操作，输入对应分页页码，进行响应操作
3. 爬取的内容全部存储在List中，并存储在request中，令前台以<s:iterator>标签迭代输出

**3.2.2 热门推荐**

1.输出热门度前八名的书籍

**热门度计算算法**

输入：用户评分分数：bookScore;

书籍对象：book.

过程：用户评分中

1: if bookScore = 3 then

2: return;

3: end if

4: if bookScore = 4 || bookScore = 5 then

5: 根据book获取喜欢该书的人数likeSize;

6: likeSize = likeSize + 1;

7: 根据book获取讨厌该书的人数dislikeSize;

8: else

9: 根据book获取喜欢该书的人数likeSize;

10: 根据book获取讨厌该书的人数dislikeSize;

11: dislikeSize = dislikeSize + 1;

12: end if

13: sumSize = likeSize + dislikeSize;

14: phat = likeSize / sumSize;

15: z = 1.96;

16: 热门度popularity = (phat + z\*z/(2\*sumSize) - z\*Math.sqrt((phat\*(1-phat) + z\*z/(4\*sumSize))) / sumSize) / (1 + z\*z/sumSize);

17: 修改该书热门度

输出：修改成功与否

注：按书籍评分，4分和5分视为喜欢，1分和2分视为讨厌

**3.3 读书社区模块**

**3.3.1 论坛**

1. 页面加载时自动调用后台程序分页输出全部推荐信息

2. 我要发帖部分使用form表单将信息传输至后台，后台进行Hibernate存储入数据库操作，图片路径存储入数据库（以二进制上传）

3. 用户对帖子进行评论时，点击发表按钮，前台用form表单将文章ID、用户对象、评论内容传至后台。后台在评论数据表中将评论内容，评论人用户名，文章ID，自动取得的评论时间存入数据库

4. 评论回复时前台Ajax将评论ID，评论内容传入后台，后台将该评论ID作为外键存入数据库

5. 赞一赞和踩一踩部分，用户点击时Ajax将type和用户对象、文章ID传入后台（赞的type为1，踩的type为2），两个部分的action相同。

6. 举报：在文章列表点击举报，将文章ID传到后台，后台根据ID取出文章对象并取出举报用户对象作为外键存储到举报数据表

**3.3.1.1 过滤垃圾文章算法**

本算法系朴素贝叶斯分类算法

输入：文章内容triContent.

过程：上传论坛文章中

1: ANSJ分词器对triContent分词，获取分词集合terms;

2: 获取terms大小size;

3: for i = 0,1,2,...,size do

4: word(i) = terms(i)的分词内容;

5: if word(i) = "http" || word(i) = "https" then

6: isExist(0) = 1;

7: else

8: isExist(0) = 0;

9: end if

10: if word(i) = "免费" || word(i) = "free" || word(i) = "赠送" then

11: isExist(1) = 1;

12: else

13: isExist(1) = 0;

14： end if

15: if word(i) = "实惠" || word(i) = "特惠" then

16： isExist(2) = 1;

17: else

18: isExist(2) = 0;

19: end if

20: if word(i) = "热卖" || word(i) = "特卖" then

21: isExist(3) = 1;

22: else

23: isExist(3) = 0;

24: end if

25: if word(i) = "国产" || word(i) = "无码" || word(i) = "porn" || word(i) = "赌场" then

26: isExist(4) = 1;

27: else

28: isExist(4) = 0;

29: end if

30: end for

31: 获取过滤表记录数allSize;

32: 获取垃圾文章数量yesSize;

33: 非垃圾文章数量noSize = allSize - yesSize;

34: rj = 1;(拉普拉斯平滑)

35: sj = 2;(第j个特征个数)

36: 垃圾文章比例p\_yes = (yesSize + rj) / (allSize + sj\*rj);

37: 非垃圾文章比例p\_no = 1 - p\_yes;

38: for i = 0,1,2,3,4 do

39: 获取isExist(i)且是垃圾文章的文章数量colation(i);

40: p\_k(i+1)\_yes = (colation(i) + rj) / (yesSize + sj\*rj);

41: 获取isExist(i)且不是垃圾文章的文章数量colation(i);

42: p\_k(i+1)\_no = (colation(i) + rj) / (noSize + sj\*rj);

43: end for

44: 是垃圾文章概率p\_litter\_yes = p\_yes \* p\_k1\_yes \* p\_k2\_yes \* p\_k3\_yes \* p\_k4\_yes \* p\_k5\_yes;

45: 非垃圾文章概率p\_litter\_no = p\_no \* p\_k1\_no \* p\_k2\_no \* p\_k3\_no \* p\_k4\_no \* p\_k5\_no;

46: if p\_litter\_yes > p\_litter\_no then

47: return 1;(是垃圾文章)

48: else

49: return 0;(非垃圾文章)

50: end if

输出：是否是论坛文章整数代号

注：计算概率时为防止分母出现为0的情况使用了贝叶斯估计法的拉普拉斯平滑。

**3.3.1.2 权威值计算算法**

输入：论坛文章ID tid;

操作类型type;

论坛文章标题triTitle;

用户对象user.

过程：赞或踩论坛文章中

1: if type = 1 then(赞)

2: 根据tid获取文章赞的数量praise

3: if (praise + 1) % 100 = 0 then

4: 根据user取出权威值userAuth;

5: userAuth = userAuth + 1;

6: 存储userAuth;

7: end if

8: else(踩)

9: 根据tid获取文章踩的数量tread;

10: if (tread + 1) % 20 = 0 then

11: 根据user取出权威值userAuth;

12: userAuth = userAuth - 1;

13: 存储userAuth;

14: end if

15: 修改赞数或踩数

过程：举报论坛文章中

1: 根据triTitle获取举报次数size;

2: if (size + 1) = 5 then

3: 根据user取出权威值userAuth;

4: userAuth = userAuth - 1;

5: 存储userAuth;

6: end if

7: 存储该举报

输出：修改成功与否

**3.3.1.3 猜你喜欢算法**

本算法系基于相似用户的协同过滤算法，核心公式为皮尔逊相似度计算公式

输入：用户对象user.

过程：赞或踩论坛文章中

1: sum1 = 0;

2: sum2 = 0;

3: sum3 = 0;

4: sum = 0;

5: 根据user获取用户全部评分对象集合list

6: 获取list大小size

7: for i = 0,1,2,...,size do

8: sum += list(i)的评分分数scoreBook(i);

9: end for

10: 用户评分平均值r1 = sum / size;

11: 获取全部用户对象集合userList;

12: userList - user;

13: 获取userList大小size;

14: for i = 0,1,2,...,size do

15: 根据userList(i)获取待测试用户评分对象集合testUserList;

16: if testUserList = 空 then

17: continue;

18: end if

19: 计算本用户和待测用户是否是相似用户，返回相同评分集合scoreList;

20: if scoreList != null then

21: 待测用户评分平均值r2(同5-10)

22： 获取scoreList(0)大小size;

23: for j = 0,1,2,...,size do

24: sum1 += (scoreList(0).(j) - r1) \* (scoreList(1).(j) - r2);

25: sum2 += (scoreList(0).(j) - r1) \* (scoreList(0).(j) - r1);

26: sum3 += (scoreList(1).(j) - r2) \* (scoreList(1).(j) - r2);

27: end for

28: if sum2 = 0 || sum3 = 0 then

29: continue;

30: end if

31: 相似度similarity = sum1 / Math.sqrt(sum2 \* sum3);

32: if similarity > 0.7 then

33: 获取待测用户收藏书籍集合list

34: end if

35: end if

36: end for

37: if list = null then

38: 根据user获取用户第一个兴趣interest;

39: 根据interest获取相同类别三本书;

40: else

41: for i = 0,1,2 do

42: 将list(i)加入结果集合resultList中;

43: end for

44: end if

输出：推荐书籍集合

**3.3.1.3.1 相似用户算法**

输入：本用户评分集合currentUserList;

待测用户评分集合testUserList.

过程：isNumSimilar(currentUserList, testUserList)

1: 获取currentUserList大小userSize

2: sizes = userSize \* 2/3;

3: num = 0;

4: for i = 0,1,2,...,userSize do

5: 获取testUserList大小size;

6: for j = 0,1,2,...,size do

7: if testUserList(j)的评分书籍 = currentUserList(i)的评分书籍 then

8: index = j;

9: end if

10: end for

11: if index != null then

12: 将currentUserList(i)的评分书籍加入currentList;

13: 将testUserList(index)的评分书籍加入testList;

14: num = num + 1;

15: if num = sizes then

16: 将currentList和testList加入scoreList;

17: return scoreList;

18: end if

19: end if

20: end for

21: return null;

输出：本用户和待测用户共同评分书籍的评分集合

**3.3.2 晒书房**

1. 使用form表单将上传信息传入后台action，并取出session中的用户对象，获取用户名，一并传到后台。后台使用Lucene引擎构建索引库并将信息存储入数据库中

2. 书籍图片路径存储入数据库（以二进制上传）

## 3.4 我的主页

**3.4.1 修改个人信息**

1. 页面加载时自动调用后台程序，取出session中的用户对象，调取数据库数据，并进行迭代输出
2. 用户名的动态验证通过Ajax进行后台传递输入的用户名，传输给用户信息的action，验证是否在数据库中存在，返回存在与否的信号
3. 点击修改按钮后，超链接跳转到后台action，将该用户ID存储在session中，result跳转至修改页面。点击确认修改后，后台取出session中的用户ID，并进行用户信息的修改

**3.4.2 我的书架**

1. 页面加载时自动调用后台程序，分页取出用户收藏数据表中全部数据
2. 点击删除按钮后使用超链接动态调用action，并将删除内容的ID传至后台，action根据ID值删除对应内容

## 3.5 管理书库

**3.5.1 修改书籍**

1. 前端在查看书籍页面中点击修改键时触发JS事件，将书籍ID存储在Cookie中，之后进入修改页面，在修改页面中用type为hidden的input标签将Cookie中的书籍ID传到后台
2. 后台根据传输过来的变量，进行是否为空和是否和原信息不同的判断，用session一级缓存进行信息修改

**3.5.2 删除书籍**

1. 同我的书架

**3.5.3 上传书籍**

1. 使用form表单将上传内容传至后台action，后台使用Lucene引擎构建索引库，并将信息存储入本地数据库
2. 书籍图片路径存储入数据库（以二进制上传）

## 3.6 管理社区

1. 同我的书架
2. 删除论坛文章时将其相应信息加入论坛文章过滤表中