长沙理工大学

《软件开发综合课程设计》

专　　业：软件工程　　班　　级：软件 2015-1 班

学生姓名：曾凯　　指导教师：傅明

起止日期：2018 年 6 月 25 日～7 月 13 日

课题名称：基于 SSH 的金句分享平台的设计与实现

实训成绩：　　老师评定：

答辩成绩：　　课程总评：

《软件开发综合课程设计》成绩评定

院　　系 计算机与通信工程院　专　　业 软件工程

班　　级 软件 2015-1 班　学　　号 _____

学生姓名 _____　完成日期 2018 年 7 月 15 日

实训成绩 _____　老师评定 _____

答辩成绩 _____　综合总评 _____

注：综合总评 = 实训成绩 * 20% + 教师评定 * 40% + 答辩成绩 * 40%，并四舍五入取整。如果缺课累计到指定次数或答辩成绩不及格，则认定综合总评为 50 分。

指导教师对学生在课程设计中的评价

评分项目	优	良	中	及格	不及格
课程设计中的创造性成果					
学生掌握课程内容的程度					
课程设计完成情况					
课程设计动手能力					
文字表达					
学习态度					
规范要求					
课程设计论文的质量					

指导教师对课程设计的评定意见

指导教师评定成绩_____指导教师签字_____ 2018 年　　月　　日

《软件开发综合课程设计》评分参考标准

一、评分子项与评分等级

按照学校发布的课程设计评分页中的 8 个评分项，根据《软件工程综合课程设计》实际，分解学校评分项为若干评分子项，并按评分等级记分（见下表 1），力求做到课程设计评分的精细化、一致行和可比性。各子项分数合计成各项分数，各项分数合计成本次课程设计成绩。

表 1、子项分与评分等级对应表

等级 / 子项分	评分等级				
	良好	中等	及格	较差	很差
5	5	4	3	2	1
4	4	3	2	1	
3	3		2	1	
2	2			1	
1	1				
0	没有做、没有考虑或没有实现				

举例：如果课题具有国内或国际领先的等级为中等，那么该子项分的最终评分为 3 分。

二、评分细则

1、课程设计中的创造性成果（10 分）

1）课题具有国内或国际领先（4 分）

2）具有自主创新的算法或应用最新的研究成果（专利）（3 分）

3）使用最新的开发工具、技术平台和运行环境（3 分）

2、学生掌握课程内容的程度（20 分）

1）提供了较好的涵盖了需求、可行性和概要设计等的方案（5 分）

2）在设计和实现中应用了软件工程的原理或方法（5 分）

3）技术路线（开发工具、运行环境等）合理可行（5 分）

4）提供了可行的测试计划（5 分）

3、课程设计完成情况（20 分）

1）按模板格式和内容提交完整的课程设计文档（5 分）

2）提供了较完整的数据字典（数据库表）（4 分）

3）完成了系统主要功能的 UI 设计或实现（4 分）

4）提供了较完整的测试用例（4 分）

4）按时提交课程设计文档（3 分）

4、课程设计动手能力（15 分）

1）数据库表设计合理满足需求（5 分）

2）UI 设计一致美观、程序代码清晰规范（5 分)

3）考虑系统出错与异常情况（5 分）

5、文字表达（5 分）

1）阐述问题准确完整（3 分）

2）文档文字通顺流畅（2 分）

6、学习态度（5 分）

1）态度端正、认真负责（3 分）

2）及时响应指导老师的通知和任务安排（2 分）

7、规范要求（15 分）

1）文档段落格式规范一致（3 分）

2）标题和正文字体大小一致（3 分）

3）图形和表格格式规范（4 分）

8、课程设计论文的质量（10 分）

1）文档摘要和目录完整准确（2 分）

2）参考文献完整正确且被引用（2 分）

3）段落和层次结构合理、处理逻辑清晰（3 分）

4）表述准确、内容通畅、结论正确（3 分）

《软件开发综合课程设计》任务书

学院：计算机与通信工程　　专业：软件工程

课程名称	软件开发综合课程设计	设计时间	2017～2018 学年 第二学期 17-19 周
学生姓名	曾凯	指导老师	傅明
题　　目	基于 SSH 的金句分享平台的设计与实现		

主要内容：进行需求分析与总体设计，确定系统需求与总体功能，然后进行详细设计与数据库设计，然后进行前端页面的编写与显示，前端页面完成后，开始使用 SSH 框架编写后台，实现了句子的发布、分类、统计、显示、喜欢、收藏、评论与回复评论，句子集的喜欢、创建，名家的喜欢、统计，出处的喜欢、统计，用户间相互关注等功能

基本要求：

1）获取与分析课题的综合需求，选择合理的技术路线，包括：系统功能结构运行平台与软件、开发工具与语言，等等。

2）应用软件工程的原理、原则、方法、步骤、技术和工具，进行系统的分析设计，合理设计总体功能结构（要有图）、，合理构建数据库（包括：ER 图、数据库表结构、库表范式等），描述系统关键算法或关键技术，编制软件测试方案，等等。

3）设计与编程实现系统的主要功能模块，按照测试方案测试软件，可以部署与运行本课题实现的软件。

4）进行课程设计总结，说明已完成工作以及未来需要重点研究与实现的内容，描述软件工程相关知识在设计开发中的应用情况。

课程设计报告：

参照《2018 年〈软件开发综合课程设计〉报告模板》的格式撰写课程设计文档，内含：背景与意义、需求与内容、技术路线、设计实现、测试方案、任务总结等等，并按封面（白色）、任务书、成绩评定页、报告正文、附录（含主要程序代码）的顺序胶装成一册。

基于 SSH 的金句分享平台的设计与实现

摘要

目前社交网络内容良莠不齐、浅薄低俗化严重，而在知乎、豆瓣等网站上关于好句子分享的帖子的热度却极高，这凸显了人们对于好句子的需求同句子分享平台的缺失之间的尖锐矛盾。目前最大的相关平台“句子迷”存在界面设计十分落后的严重缺陷，难以吸引新用户的加入。基于上述考虑，本项目决定制作一个金句分享平台，供人们分享私藏的好句子，帮助人们提高文学素养。

本文从本系统开发等背景开始，以系统功能模块以及软件设计作为主要陈述内容，包含了需求分析，总体设计以及详细设计三大部分。系统后台采用 tomcat 作为服务器，mysql 作为数据库，利用 SSH 框架作为后端应用框架。前台开发使用 sublime Text 3 进行开发，后台开发使用 IDEA 开发工具。同时使用 DWR 框架提高用户与页面之间交互体验。经过了设计、实现与测试，本系统已基本完成系统的预期功能。

关键词：金句分享平台；SSH；mysql；DWR

DESIGN AND IMPLEMENTATION OF PROGRAM – BASED SSH GOOD SENTENCE SHARING WEBSITE

ABSTRACT

At present, the content of social networks is mixed, shallow and vulgar, but the popularity of posts about good sentence sharing on websites such as Zhihu and Douban is extremely high, which highlights the sharp contradiction between the need for good sentences and the lack of sharing platforms. The largest relevant website "Juzi Mi" has serious flaws in interface design, which is difficult to attract new users. Based on the above considerations, the project decided to create a platform for sharing sentences, provide people a place to share good sentences and to help people improve their literacy.

This paper begins with the background of the system development, with the system function module and software design as the main content, including the requirements analysis, overall design and detailed design. The system background uses tomcat as the server, mysql as the database, and the SSH framework as the back-end application framework. The foreground development uses sublime Text 3 for development, and the background development use IDEA development tools. At the same time, the DWR framework is used to improve the interaction between the user and the page. After design, implementation and testing, the system has basically completed the expected function of the system.

Key words: good sentences sharing platform; SSH; mysql; DWR

目 录

1 绪论

近些年里，生活压力的不断增大，很多人们已经失去看书的习惯，从前以看书为消遣的时代已经过去，现在人们的转而去刷段子、看直播和短视频，以图一时之乐，人文素养已经开始逐渐下降，网络舆论的娱乐化传播给社会造成了不少负面影响[1]。可是从知乎等网站的帖子里可以看出人们对于好句子的需求是非常大的[2]，可是这种网站中的句子都是零星散布，没有得到统一的整理与维护，也不能针对具体句子进行搜索与延伸，而国内句子分部平台巨头“句子迷”却因为界面风格过于老旧而广受诟病，国内需要一个内容丰富、功能完整、界面美观、操作简单、方便高效的句子发布与分享网站。

1.1 课题研究的背景

生活节奏加快、生活压力增大，使得人们时间变得碎片化、精神变得越来越疲乏，人们大部分空闲时间花在了社交网络上，而近年来，社交网络上的信息开始倾向于娱乐化、低俗化、肤浅化，特别是近年来网络直播、短视频等丑闻频出，网络泛娱乐化已经在不断挑战社会底线。而普通人们对于文学的关注程度显然远远不及对娱乐信息的关注程度，但是文学对于一个人的思想道德修养的益处是无穷的，特别是一些意韵深远的句子与短诗，这也让很多人开始热衷于好句子的收集与抄写，但是却缺少一个好的查询与分享句子的平台，而这正是本网站所要解决的问题。

1.2 课题的研究意义

在信息化时代，句子的分享与收藏方式必然需要集中在网上，以前的那种好句子手抄本的时代已经很难重现了，而截至 2017 年 12 月，我国网民数量已经达到 7.7 亿，普及率至 55.8%，同时由于以智能手机为代表的移动通信终端的

普及，本网站也相应做了适配手机端的响应式布局，以使用户无论是通过手机还是电脑均可很方便访问本网站，这样就可以充分利用用户碎片化空余时间，也可以方便用户使用，尽量给用户提供最好的用户体验。

1.3 国内外现状与发展趋势

国内关于好句子分享的正规网站只有“句子迷”一家，但是界面设计并不美观，而且并不支持响应式设计，其余类似网站除了上述问题外还存在句子数量较少、广告多等问题，而国外相对而言，并没有专门的好句子分享网站，即便是一些名言的分享也都比较老旧。博大精深的中华文化的网络传播存在严重缺乏的现象，这不利于中国青少年的文化素养的培育和发展， 在经济水平高速发展的同时，思想文化建设必须要齐头并进，而这一点在网络上面的传播并未得到相关部门的重视，不过未来应该会有所改善。

1.4 论文结构和内容

第一章，结合人们对于好句子的需求与国内网络内容泛娱乐化的现状进行分析，揭示出好句子分享网站的缺失和不足，并分析其在国内外发展现状与趋势。

第二章，介绍本次项目所运用到的主要、核心技术，以及他们各自的特点。

第三章，利用场景分析等软件工程需求分析方法[8]，进行项目各个功能模块的进行需求分析。

第四章，通过 E-R 图、数据流图等图表与文字来陈述本次总体设计和数据库设计。

第五章，详细叙述本系统各个功能模块具体的功能以及实现结果。

2 相关技术简介

本系统开发前端采用 html、CSS、javaScript 进行开发，后端采用 java 进行开发，后端服务容器采用的是 tomcat 6.0.20，网页适配 Chrome、FireFox、Safari、Edge 等主流浏览器高版本。

2.1 Spring Framework

Spring Framework 是由 Rod Johnson 博士于 2004 年 3 月 24 日正式发布，它目前是 Java Web 开发中的主流框架之一。其具备轻量、控制反转、面向切面、容器、框架、MVC 等特征，这些特征能够使编写的程序更干净、更可管理、并且更易于测试。它们也为 Spring 中的各种模块提供了基础支持。

Spring 框架的作用是整合，不仅是对项目内部的整合，也支持了和一切其他主流框架的整合。其特点在于解开耦合、简化开发、支持 AOP 编程、支持声明式事务、方便程序的测试、方便继承各种优秀框架、降低 Java EE API 的使用难度[3]。其具有以下优点：

1. 低侵入式设计，代码污染极低；
2. 独立于各种应用服务器，基于 Spring 框架的应用，可以真正实现“Write Once, Run Anywhere”的承诺；
3. Spring 的 DI 机制降低了业务对象替换的复杂性，提高了组件的解耦；
4. Spring 的 AOP 支持允许将一些通用任务如安全、事务、日志等进行集中式管理，从而提供了更好的复用；
5. Spring 的 ORM 和 DAO 提供了与第三方持久层框架的良好整合，并简化了底层的数据库访问；
6. Spring 并不强制应用完全依赖于 Spring，开发者可自由选用 Spring 框架的部分或全部

2.2 Struts2

Struts2 是一个基于 MVC 设计模式的 Web 应用框架，是 Struts 的下一代产品，是在 struts 1 和 WebWork 的技术基础上开发而来的。它本质上相当于一个 servlet，在 MVC 设计模式中，Struts2 作为控制器(Controller)来建立模型与视图的数据交互。Struts 2 的体系结构与 Struts 1 的体系结构差别巨大，以 WebWork 为核心，采用拦截器的机制来处理用户的请求，这样的设计也使得业务逻辑控制器能够与 ServletAPI 完全脱离开，所以 Struts 2 可以理解为 WebWork 的更新产品[4]。以下是其优点：

1. 实现了 MVC 模式，层次结构清晰，使程序员只需关注业务逻辑的实现；
2. 丰富的标签库，大大提高了开发的效率；
3. Struts2 提供丰富的拦截器实现；
4. 通过配置文件，就可以掌握整个系统各个部分之间的关系；
5. 异常处理机制，只需在配置文件中配置异常的映射，即可对异常做相应的处理；
6. Struts2 的可扩展性高；
7. 面向切面编程的思想在 Strut2 中也有了很好的体现。

2.3 Hibernate

Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，它将 POJO 与数据库表建立映射关系，是一个全自动的 orm 框架，hibernate 可以自动生成 SQL 语句，自动执行，使得 Java 程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用，最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任[5]。其有如下优点：

1. 对象化。人员以面相对象的思想来操作数据库。Hibernate 支持许多面向对象的特性，如组合，继承，多态等；

2. 更好的移植性.对于不同的数据库，开发者只需要使用相同的数据操作即可；
3. 开发效率高。Hibernate 提供了大量的封装，很多数据操作以及关联关系等都被封装的很好，开发者不需写大量的 sql 语句，极大的提高了开发者的开发效率；
4. 缓存机制的使用。Hibernate 提供了缓存机制（session 缓存，二级缓存，查询缓存），对于那些改动不大且经常使用的数据，可以将它们放到缓存中，不必在每次使用时都去查询数据库，缓存机制对提升性能大有裨益。

2.4 DWR

DWR（Direct Web Remoting）是一个用于改善 web 页面与 Java 类交互的远程服务器端 Ajax 开源框架，可以帮助开发人员开发包含 AJAX 技术的网站。它可以允许在浏览器里的代码使用运行在 WEB 服务器上的 JAVA 方法，就像它就在浏览器里一样。

DWR 采取了一个类似 AJAX 的新方法来动态生成基于 JAVA 类的 JavaScript 代码。这样 WEB 开发人员就可以在 JavaScript 里使用 Java 代码，就像它们是浏览器的本地代码(客户端代码)一样；但是 Java 代码运行在 WEB 服务器端而且可以自由访问 WEB 服务器的资源。出于安全的理由，WEB 开发者必须适当地配置哪些 Java 类可以安全的被外部使用[6]。

DWR 框架有如下优点：

1. 实现了浏览器兼容；
2. 进行了 json 的封装;
3. 可以直接调用 java 中 service 的方法（看似如此）;
4. 可以少写很多 js 代码；
5. 可以实现消息的推送；

3 需求分析

3.1 设计目标

本系统的目的，首先是要完成好句子的发布、评论、收藏、分类、标签，并进行名家和作品的喜欢，用户间的关注等等；其次是要考虑到用户的使用体验，要做到响应式布局，让用户在手机上也能方便使用；最后是要保证页面内变动要做到平滑变化，尽量使用 DWR 实现异步操作，避免频繁页面跳转带来的用户体验下降。

3.2 可行性分析

从系统开发的经济、技术、社会、法律以及其他方面，分析系统开发的可行性，综合各个因素，整体评估系统是否能在现有的条件上得到实现，并且是否合算，从而确定整个系统的可行性。

3.2.1 经济可行性

本网站以方便人们分享与查找好句子为核心，坚持培养当代人们的人文情怀，但是作为一个商业网站，其盈利方式不能太过直白，也不能有所欠缺。本项目在前期发展阶段不打算进行盈利，等用户基数达到一定程度后开始推出售卖网站周边、举办系列相关活动（收取小额费用）、出售低额网课、句子推出打赏功能（收取手续费）、往页面投放少量广告、开通微信公众号并在公众号中推送广告等等。由于本网站运营和维护成本低，且对于用户有情怀感，因此经济上完全可行。

3.2.2 技术可行性

本网站所使用的的技术都比较成熟，是在许许多多框架里被挑出来的优秀框架，DWR 和 SSH 框架的社区都十分活跃，网上关于它们的资料众多，技术

上完全可行。

3.2.3 操作可行性

本网站设计同一班网站类似，熟悉常用计算机操作的用户均可迅速熟悉本网站的使用；而对于手机端，也进行了相关响应式适配，用户在手机上也可以如在电脑上一样访问本页面并进行操作，因此本网站具有操作可行性。

3.2.4 其他可行性

系统为课程设计课题，由学生个人在指导教师的指导下按照软件开发的流程，进行设计开发。属于学生的自主设计和开发行为，整个开发过程，未侵犯他人专利技术，系统内容规范健康，不涉及不良信息，符合法律要求。涉及的虚拟销售产品和测试数据健康积极，符合社会规范，符合道德可行性。

3.3 性能需求

本网站是一个好句子分享网站，数据量较大，因此对于存储效率、响应速率均有较高要求。

1. 存储效率

本网站采用 hibernate 进行数据库存取操作，数据表的设计并未完全消除冗余，如此考虑是为了实现数据的快速读取，尽可能降低读取数据的时间负担。

2. 响应速率

本网站为结局响应速率问题，部分页面采取了数据的异步加载与本地缓存机制，部分页面对于数据量进行了限制，尽量保证所有网页均能在 3 秒内加载完毕。

3. 界面友好性

本网站的界面采用的是一般网站常用的布局格式，用户易于上手，同时页面上显示的内容。

3.4 功能需求

本系统所应该包含的必要的三大功能模块有：发布句子、信息统计、访问限制。

3.4.1 发布句子

用户可以发布句子，发布句子可以选择是否是原创，并且在发布句子时可以将句子存入自己的句子集中，且如果句子为非原创，则可以输入出处和句子的作者，而如果数据库中有相关出处和作者，那么会将该句子自动添加到其下，如果数据库之前没有相关出处和作者，那么会自动创建该作者和出处。

3.4.2 信息统计

信息统计包含的内容非常多。对用户而言，包括用户的资料编辑、用户喜欢的句子、用户发布的句子、用户发布的原创句子、用户创建的句子集、用户关注的句子集、用户关注的用户、用户关注的名家、用户关注的出处、用户的粉丝等；对句子而言，包括句子信息、该句子对应的评论信息、喜欢该句子的用户列表、该句子的作者和出处信息等；对句子集而言，有句子集信息、该句子集包含的句子列表；对名家而言，有名家自身资料、名家的句子列表、喜欢该名家的用户列表、名家作品列表等；对出处而言，有出处自身信息、出处包含的句子列表、喜欢该出处的用户列表等；。

3.4.3 访问限制

本网站部分页面的访问是需要权限的，比如修改资料，则必须是得登录了才能访问，否则会跳转到登录页。同样，如果未登录便进行喜欢句子、收藏句子、喜欢名家、喜欢出处等，则会提示先进行登录才能进行操作。

4 总体设计

在本章节中陈述的是本系统设计的具体功能模块的层次结构以及数据库相关的设计。

4.1 概要设计

本系统的具体大模块有如下四个：基本操作、信息统计、访问限制和信息管理。其功能模块图如 4.1 所示

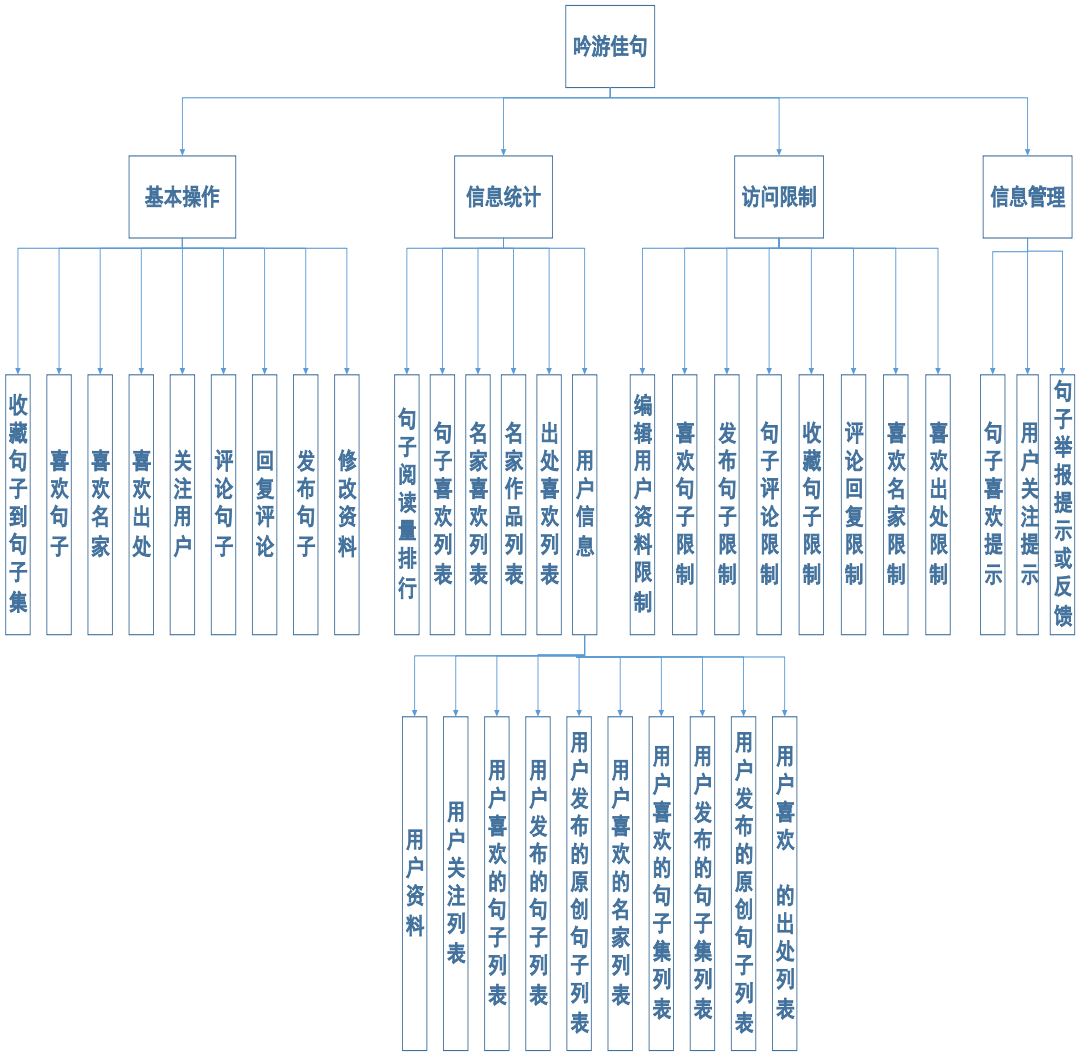

图 4.1 金句分享平台总体设计图

4.2 数据库设计

4.2.1 数据库结构设计

为了更加直观地表示概念模型，一般常用概念模型来进行表示。表示方法有很多，其中 E-R 方法是 PPSChen 于 1976 年提出的实体联系方法（Entity-Relationship approach）。该方法用 E-R 图来描述现实世界的概念模型[7]。

根据数据库及功能要求，我们可以对整个系统的各个实体与其之间的联系有一个打给的结构。图 4.2 为系统 E-R 图。

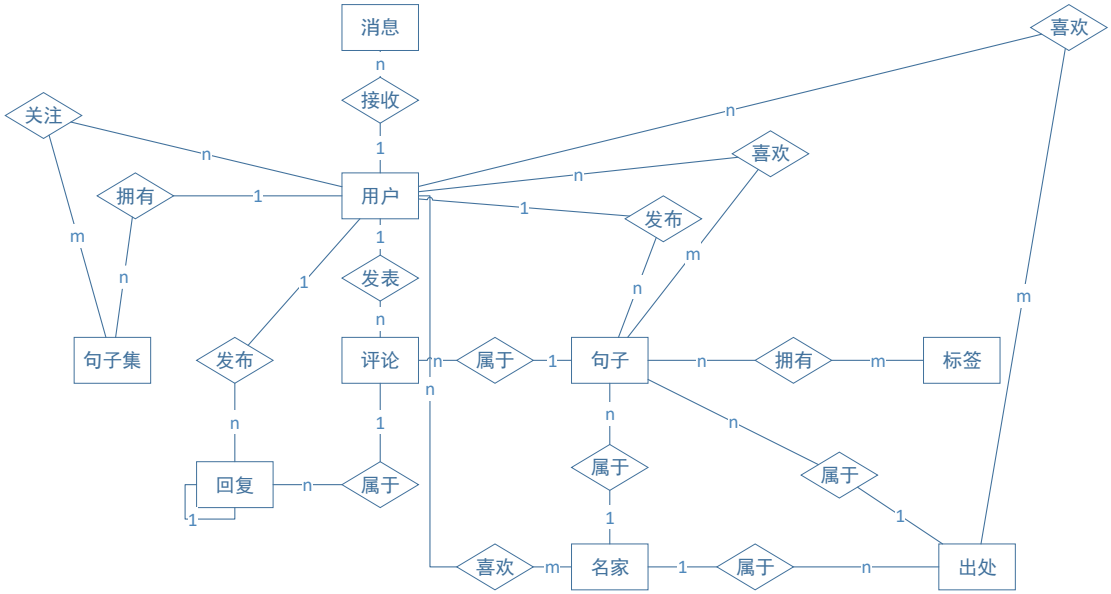

图 4.2 系统 E-R 图

下面介绍的是每个实体的详细属性。如图 4.3 所示，为用户信息实体的属性。本系统为了提高用户查找安全性，将用户信息实体拆分为数据库中两个表，一个为登录注册的信息，另一个为用户详细信息，同时在注册时也分两个页面进行输入。

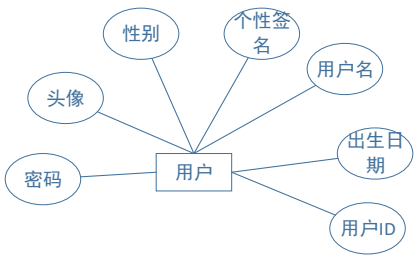

图 4.3 用户信息实体图

句子集实体如图 4.4 所示。每一个句子集包括句子集自身信息表、句子集包含的句子表，以及喜欢句子集的用户表。

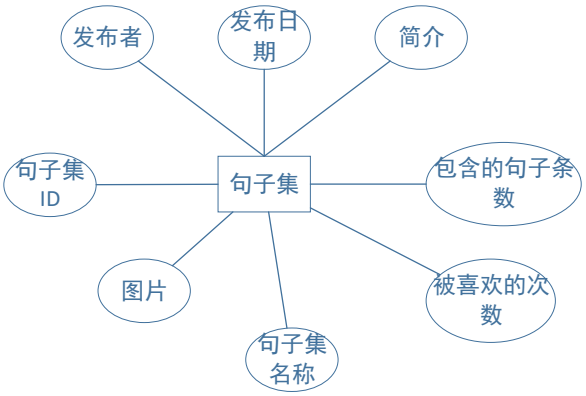

图 4.4 句子集属性

句子实体如图 4.5 所示。其包含的是一条句子所含有的主要信息，其中标签是以字符串形式存储，而标签是会被后台自动分割存到另外的表中的。

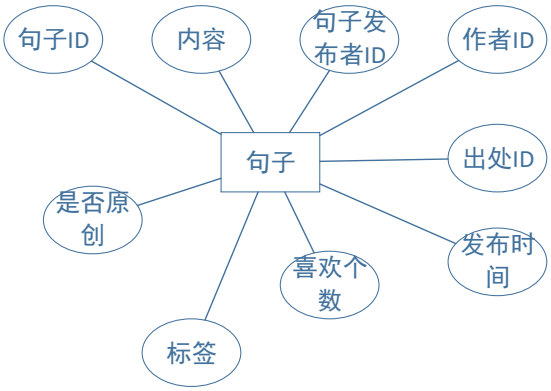

图 4.5 句子属性

标签实体如图 4.6 所示。其包含的是句子的标签信息，每一个句子在发布的时候可以设置标签，最多设置 5 个。

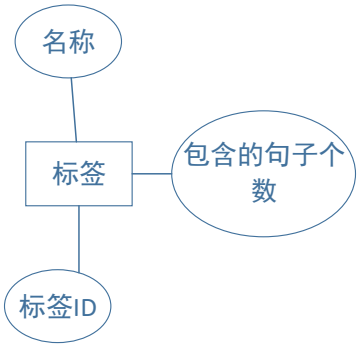

图 4.6 标签属性

名家实体如图 4.7 所示。名家即为非原创的句子的作者，可以不填，在发布句子的时候如果该名家不存在于数据库则会自动创建。

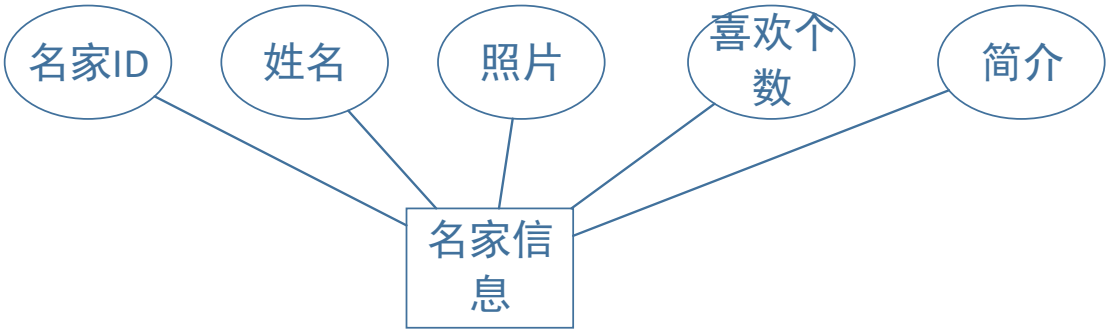

图 4.7 名家属性

出处实体如图 4.8 所示。出处及非原创句子的出处，可以不填，同名家一样，也会在发布句子时，如果当前出处不存在于数据库，则会自动创建。不过如果同时填了名家和出处，那么该出处还会自动算入名家之下。

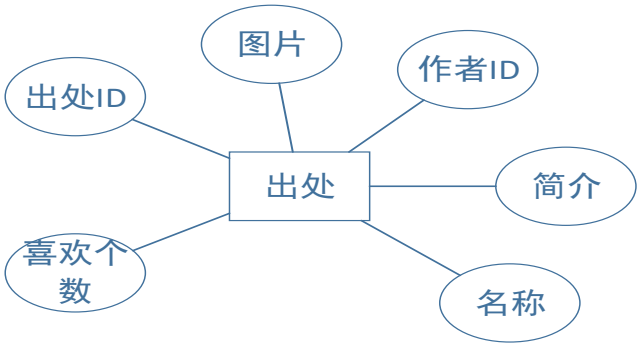

图 4.8 出处属性

消息实体如图 4.9 所示。包含了用户收到的所有信息，包括动态信息与举报信息等。

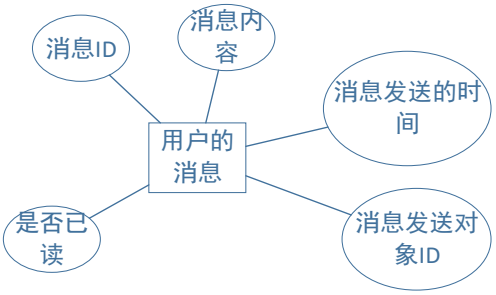

图 4.9 消息属性

评论实体如图 4.10 所示。用户在

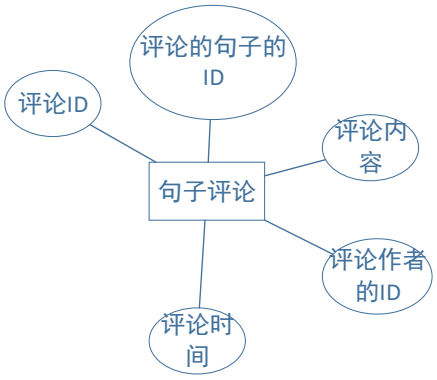

图 4.10 评论属性

回复实体如图 4.11 所示。回复和评论是分开的，评论挂在句子下面，回复挂在评论下面，这样方便分别管理。每个回复只需保存其所回复的评论或回复的 ID 即可确定其位置。

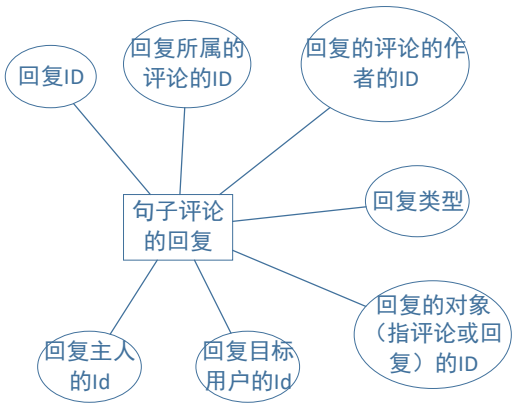

图 4.11 回复属性

4.2.2 数据库表结构设计

根据上文对本系统对数据流分析和数据库实体分析，可以建立数据库物理结构。本节将对数据库表结构设计进行叙述，建表规则参考了阿里巴巴 Java 开发手册[10]。

(1)用户登录信息表

登录信息包含用户 Id 及用户名和密码，以供用户登录之用，如表 4-1 所示。

表 4-1 用户登录信息表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	用户编号，自增长
name	VARCHAR	20	否	否	用户名（唯一）
pwd	VARCHAR	16	否	否	用户密码（长度 6-16 位）

(2)用户信息表

表 4-2 所示为用户信息表，用于存储用户的详细信息，如表 4-2 所示。

表 4-2 用户信息表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	用户编号，自增长
user_name	VARCHAR	20	否	否	用户名
head_path	VARCHAR	255	否	否	用户头像路径（提供默认头像）
gender	CHAR	1	否	否	用户性别
birth	Date	32	否	否	用户出生日期,默认为 2000-01-01
motto	VARCHAR	64	否	可	个性签名

(3)句子集表

句子集信息如下表 4-3 所示，其包含了句子集的常规属性，以及句子集包含的句子数量和喜欢句子集的用户的数量，如表 4-3 所示。

(4)句子表

句子表为所存储的句子的信息，如表 4-4 所示。

(5)标签表

标签表包含了句子的标签名及标签被引用的次数信息。其表结构如表 4-5 所示。

表 4-3 句子集表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	句子集编号，自增长
name	VARCHAR	25	否	否	句子集名称，允许重复
publisher_id	Unsigned bigint	20	否	否	发布者 ID
publish_date	Datetime	32	否	否	发布日期
img_path	VARCHAR	255	否	否	句子集图片路径，默认为用户头像
introduction	VARCHAR	255	否	可	句子集介绍，默认为空
sentence_num	Unsigned bigint	20	否	否	句子集里包含的句子个数
love_num	Unsigned bigint	20	否	否	喜欢该句子集的人的个数

表 4-4 句子表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	句子编号，自增长
content	VARCHAR	255	否	否	句子内容，允许重复
publisher_id	Unsigned bigint	20	否	否	发布者 ID
is_original	Unsigned tinyint	3	否	否	是否原创，1 表示是，0 表示不是
author_id	Unsigned bigint	20	否	可	作者 ID，可不填
origin_id	Unsigned bigint	20	否	可	出处 ID，可不填
publish_time	Datetime	32	否	否	句子发布时间
love_num	Unsigned bigint	20	否	否	喜欢该句子的人的个数
tag	VARCHAR	16	否	否	的库存数数

表 4-5 标签表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	标签编号，自增长
name	VARCHAR	25	否	否	标签名称，不允许重复
quote_num	Unsigned bigint	20	否	否	该标签被引用数量，初始化为 1(因为新增一个标签，意味着有一个句子使用了该标签)

(6) 名家信息表

名家信息表里包含一个名家自身的基础信息，以及喜欢该名家的数量，每当有人喜欢或者不喜欢该名家时其数量会自动变化。其表结构如表 4-6 所示。

表 4-6 名家信息表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	名家编号，自增长
name	VARCHAR	25	否	否	名家名称，不允许重复
img_path	VARCHAR	255	否	否	名家照片地址
introduction	TEXT		否	否	名家介绍
love_num	Unsigned bigint	20	否	否	喜欢该名家的个数

(7) 出处信息表

出处信息表包含了出处的信息和出处被喜欢的次数。其表结构如表 4-7 所示。

(8) 消息表

消息表里记录了用户收到的消息，包括是否阅读、内容、消息接收者、发送时间等内容。其表结构如表 4-8 所示。

表 4-7 出处信息表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	出处编号，自增长
name	VARCHAR	25	否	否	出处名称，不允许重复
img_path	VARCHAR	255	否	否	名家照片地址
author_id	Unsigned bigint	20	否	可	出处的作者的 ID
introduction	TEXT		否	可	出处介绍
love_num	Unsigned bigint	20	否	否	喜欢该出处的个数

表 4-8 消息表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	消息编号，自增长
content	VARCHAR	500	否	否	消息内容
send_time	Datatime	32	否	否	消息发送时间
send_user_id	Unsigned bigint	20	否	否	这条消息要发送给的目标用户
is_read	Unsigned tinyint	20	否	否	消息是否已读，1 为是，0 为否

(9) 句子评论表

句子评论表包含的是句子的评论信息。其表结构如图 4-9 所示。

表 4-9 句子评论表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	句子评论编号，自增长
content	VARCHAR	255	否	否	评论内容
sentence_id	Unsigned bigint	20	否	否	评论所属句子 ID
user_id	Unsigned bigint	20	否	否	发送这条评论的用户的 ID
comment_time	Datetime	32	否	否	进行评论的时间

(10) 回复表

回复表是针对于每一条评论下的回复设计的，可能是回复评论，也可能是回复回复。其表结构如图 4-10 所示。

表 4-10 回复表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	回复的 ID
content	VARCHAR	500	否	否	回复的内容
comment_id	Unsigned bigint	20	否	否	回复所在的评论的 ID
reply_type	Unsigned tinyint	20	否	否	回复的类型，0 为回复评论，1 为回复回复
reply_comment_user_id	Unsigned bigint	20	否	否	回复所在的评论的作者的 ID
reply_object_user_id	Unsigned bigint	20	否	否	回复的用户的 ID，如果是回复评论，那么就是评论作者 ID，如果回复的是回复，那么就是回复的回复的作者的 ID
reply_object_id	Unsigned bigint	20	否	否	被回复的那一条评论或回复的 ID，如果 reply_id 为 0，这里就是评论 ID，否则为回复 ID
reply_writer_id	Unsigned bigint	20	否	否	发送这条回复的用户的 ID
reply_time	Datetime	32	否	否	进行回复的时间

(11) 句子喜欢表

句子喜欢表包含的是喜欢句子的表的信息。其表结构如图 4-11 所示。

(12) 句子集喜欢表

句子集喜欢表包含的是喜欢句子集的表的信息。其表结构如图 4-12 所示。

表 4-11 句子喜欢表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
user_id	Unsigned bigint	20	否	否	用户 id
sentence_id	Unsigned bigint	20	否	否	句子 id

表 4-12 句子集喜欢表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
user_id	Unsigned bigint	20	否	否	用户 id
collection_id	Unsigned bigint	20	否	否	句子集 id

(13) 名家喜欢表

名家喜欢表包含的是喜欢名家的表的信息。其表结构如图 4-13 所示。

表 4-13 名家喜欢表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
user_id	Unsigned bigint	20	否	否	用户 id
giant_id	Unsigned bigint	20	否	否	名家 id

(14) 出处喜欢表

出处喜欢表包含的是喜欢出处的表的信息。其表结构如图 4-14 所示。

表 4-14 出处喜欢表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
user_id	Unsigned bigint	20	否	否	用户 id
origin_id	Unsigned bigint	20	否	否	出处 id

(15) 标签引用表

标签引用表包含的是标签的引用的表的信息。其表结构如图 4-15 所示。

表 4-15 标签引用表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
sentence_id	Unsigned bigint	20	否	否	句子 id
tag_id	Unsigned bigint	20	否	否	标签 id

(16) 句子集-句子表

句子集-句子表包含的是句子集中句子的信息。其表结构如图 4-16 所示。

表 4-16 句子集-句子表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
collection_id	Unsigned bigint	20	否	否	句子集 id
sentence_id	Unsigned bigint	20	否	否	句子 id

(17) 类别表

类别表包含的是类别的信息，普通用户不能对其进行增删改操作。其表结构如图 4-17 所示。

表 4-17 类别表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
category_name	Unsigned bigint	20	否	否	句子类别名称，唯一

(18) 句子-类别表

句子-类别表包含的是句子与类别间的信息。其表结构如图 4-18 所示。

表 4-18 句子-类别表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
sentence_id	Unsigned bigint	20	否	否	句子 ID
category_id	Unsigned bigint	20	否	否	句子类别 ID

(19) 用户间关注表

用户间关注表包含的是用户间关注的信息。其表结构如图 4-19 所示。

表 4-19 用户间关注表

字段名	数据类型	字段长度	是否为主键	可否为空	说明
id	Unsigned bigint	20	是	否	唯一主键，自增长
user_id	Unsigned bigint	20	否	否	用户 ID
follower_id	Unsigned bigint	20	否	否	粉丝 ID

5 详细设计与系统实现

在本文上述章节中已经进行了本系统的需求分析和概要设计。在此基础上，对本系统进行详细设计与系统实现。

5.1 开发运行环境

操作系统：windows 10

开发语言：html，css，js，java

数据库：mysql 5.7.14

开发工具：sublime 3.0，IDEA 2017.2 等

应用服务器：Tomcat 6.0.20

服务器端框架：Spring，Struts2，Hibernate，DWR

5.2 系统流程图

发布句子是系统核心功能之一，其实现的功能是用户点击发布句子按钮，判断用户是否登录，如果未登录则提示进入登录页，否则进入句子内容填写页，用户填好句子信息后点击确认，如果提交成功，则进入句子详情页，否则跳转错误页。发布句子流程图如图 5.1 所示。

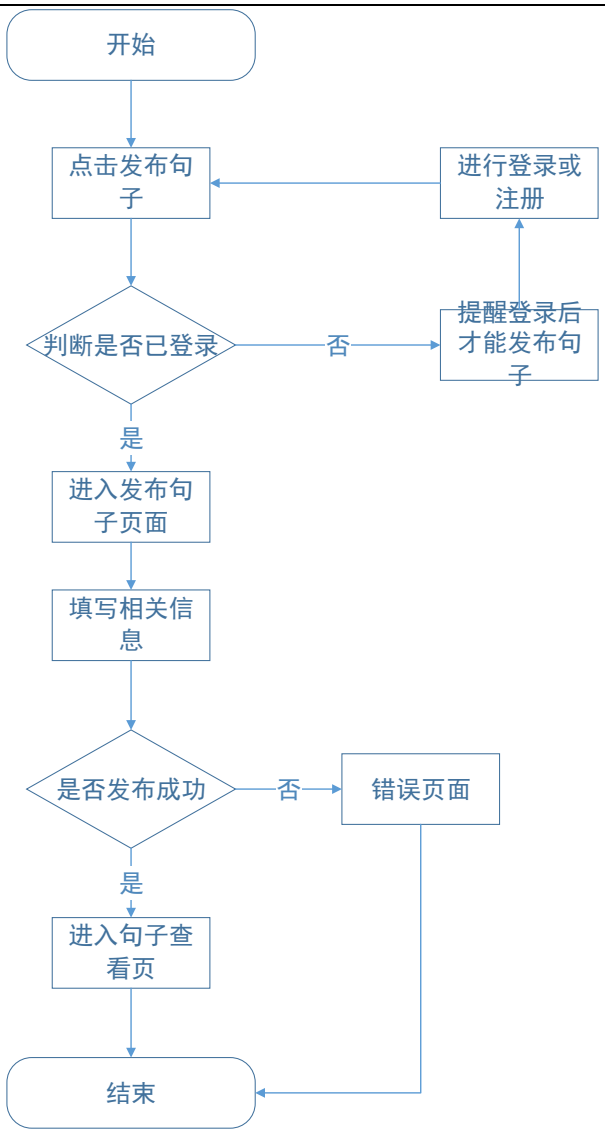

图 5.1 发布句子功能流程图

句子评论与回复也是系统的主要功能之一，用户点击评论按钮或者回复按钮即可进行评论或回复，但是当点击发布的时候，会检查用户是否登录，如果用户没有登录，那么会提示用户先登录后才能发表评论或回复。句子评论与回复流程图见图 5.2。

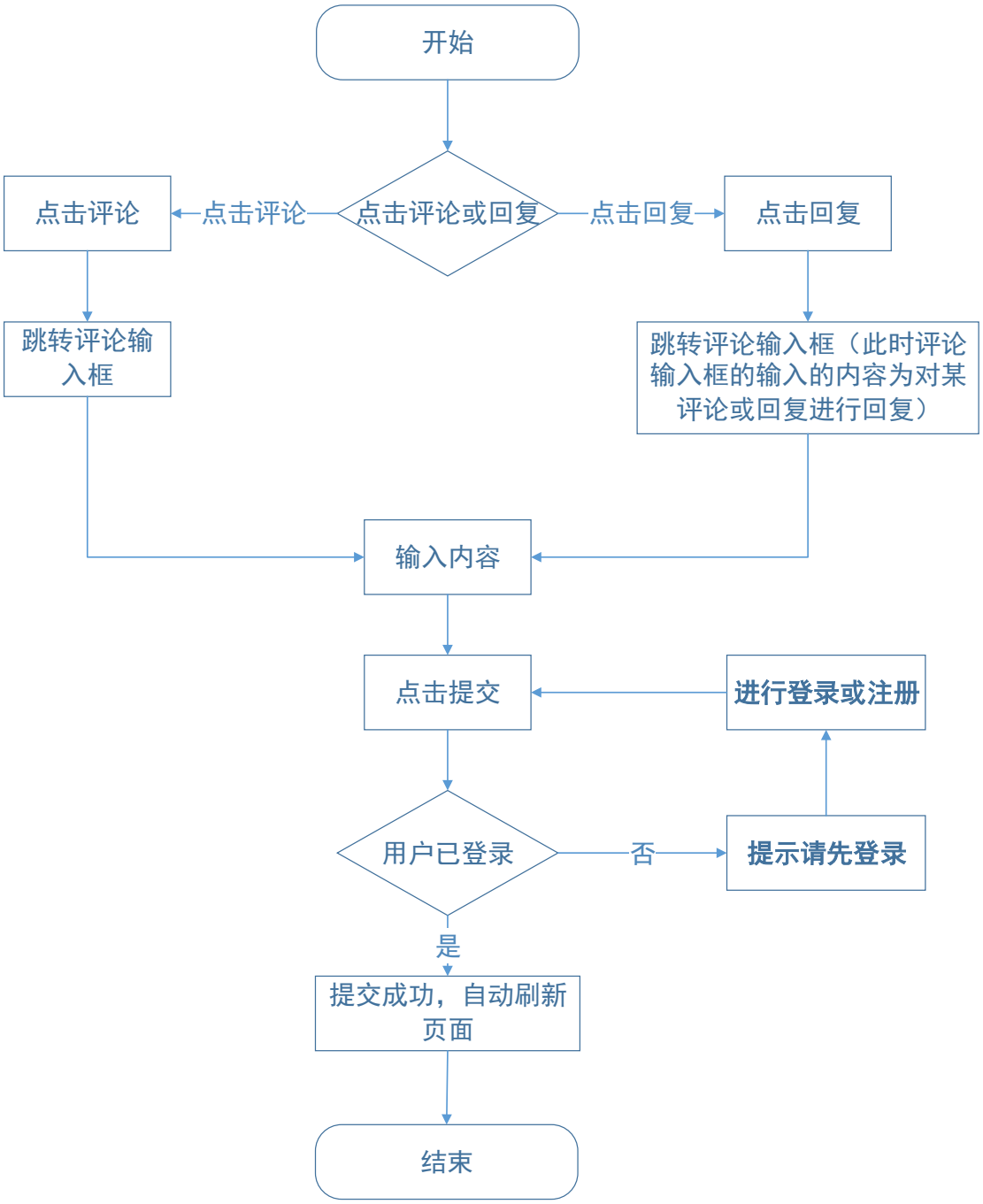

图 5.2 句子评论与回复功能流程图

5.3 运行情况图

本网站基本功能已得到基本实现，下面放上一些网站运行截图。

网站首页如图 5.3 所示，顶部为随机句子（每次打开页面都会随机生成

一条），另外还有热门名人、推荐句子、热门句子、热门原创、最新发布等分类，还有发布句子、搜索等功能按钮，以及句子的分类、标签，以及热门句子集。

图 5.3 网站首页

如图 5.4 为句子发布页，用户可以根据自身所想发布的句子的情况填入相应信息。

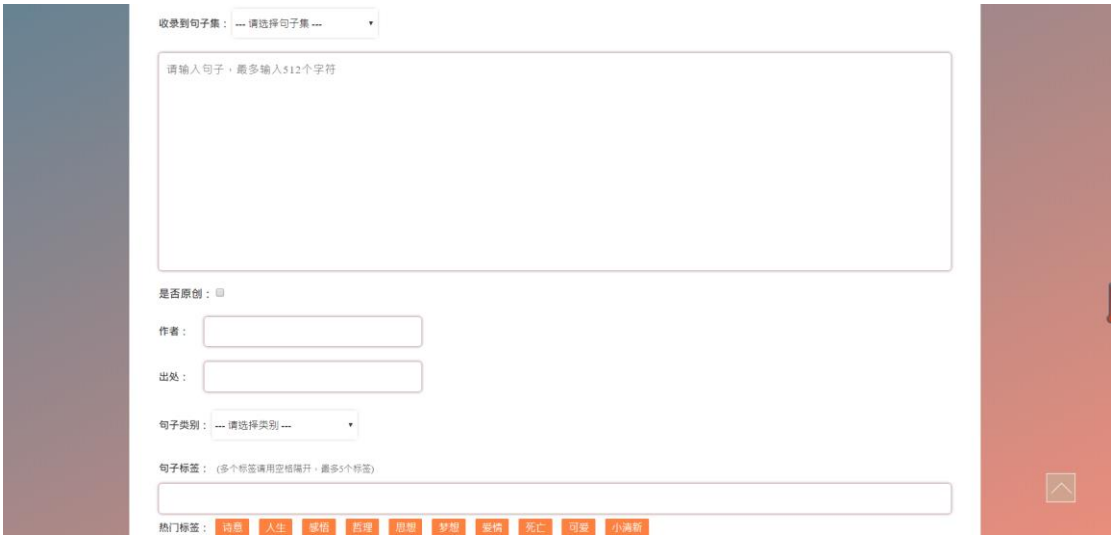

图 5.4 句子发布页

图 5.5 为句子详情页，其除了显示句子一般信息外，还可以进行评论、显示名家信息、以及喜欢该句子的用户等等。

图 5.5 句子详情页

图 5.6 为名家信息页，包含名家介绍、喜欢该名家的用户、名家的作品信息等。

图 5.6 名家信息页

图 5.7 为名家句子页，包含了名家的简单介绍（只显示三行），以及名家代表作、以及句子列表等。

图 5.7 名家句子页

图 5.8 为出处信息页，包含了出处的信息、作品的介绍、喜欢该出处的用户、以该出处包含的句子。

图 5.8 出处信息页

图 5.9 为个人信息页，拥有用户信息、喜欢的句子、发布的句子、原创的句子、句子集、喜欢的句子集、关注着、粉丝、关注的名家和出处等信息。

图 5.9 出处信息页

图 5.10 为句子集信息表，包含了句子集的创建者、创建时间、名称等基本信息，以及包含的句子信息。

图 5.10 句子集信息页

6 测试

6.1 测试目的

尽管在软件工程科学的方法指导下完成了系统的设计与开发，但是由于软件系统是一个关系紧密且复杂的逻辑系统，因此仅凭严格的设计以及严格的开发流程并不能够完全确保系统不会出现任何缺陷。不存在缺陷的系统是不可能存在的，但是通过软件测试，我们能够尽可能多的在软件系统投入使用前发现目前存在的缺陷并对缺陷进行修复。软件测试与维护这个生命周期是软件生命周期中最长的亦是最重要的，这不仅保证了系统的稳定性以及满足预定的需求，也能够使得软件生命得到延续。因此，通过软件测试对系统进行评估，找到其中隐藏的缺陷并对其加以修复是很有必要的。

软件测试是为了完善系统质量的技术，通过翻阅书籍，发现大多对软件测试的定义为使用人工或自动的手段来运行某个系统的过程，其目的在于检验它是否满足规定的需求，或是弄清预期结果与实际结果的区别。

6.2 测试用例

6.2.1 发布句子测试

本功能为系统的主要功能，为句子分享的核心部分。具体的测试如表 6-1 所示。

表 6-1 发布句子测试用例

前提	测试步骤	预期结果	实际结果	是否通过
用户进入网站首页	1. 未登录时点击发布句子	提示“请先登录呦”	提示“请先登录呦”	通过
	1. 登录进系统 2. 点击发布句子	进入发布句子页	进入发布句子页	通过
	1. 登录进系统 2. 点击发布句子 3. 填写句子信息 4. 填写句子作者信息为鲁迅 5. 选择为原创	句子显示为原创	句子显示为原创	通过
	1. 登录进系统 2. 点击发布句子 3. 正确填写句子信息 4. 选择收藏到句子集（“吟游句子集”）	该句子被收藏到吟游句子集	该句子被收藏到吟游句子集	通过

6.2.2 用户评论测试

在用户评论中可能出现注入攻击，有可能可以在里面写 css 代码以改变页面样式，下面对其进行测试。具体测试情况如表 6-2 所示。

表 6-2 用户评论用例

前提	测试步骤	预期结果	实际结果	是否通过
用户进入某句子详情页	1.未登录时点击发表评论	提示“请先登录呦”	提示“请先登录呦”	通过
	1. 用户已登录 2. 不输入评论，直接点击发布	提示“请输入内容后再提交呦”	提示“请输入内容后再提交呦”	通过
	1. 用户已登录 2. 评论全为空格，直接点击发布	提示“请输入内容后再提交呦”	提示“请输入内容后再提交呦”	通过
	1. 用户已登录 2. 输入吟游佳句	评论显示“吟游佳句”	评论显示“吟游佳句”	通过

6.3 测试结果分析

本系统总体上通过了主要基本功能模块的测试，但是正如前文所说的那样，仍然还是不能够保证本系统是毫无缺陷的，但是通过本次测试使得本系统原本不合理的设计得到修改。尽管如此，本系统在安全性以及稳定性上仍有欠缺之处，需要日后进一步完善。

参考文献

[1] 邓向阳, 向娴. 网络舆论娱乐化传播的负面社会效应[J]. 青年记者, 2018(3):37-38.

[2] 知乎网. 有哪些惊艳到你的句子？[DB/OL]. https://www.zhihu.com/question/55962172

[3] 百度百科. Spring [DB/OL] https://baike.baidu.com/item/spring/85061?fr=aladdin

[4] 百度百科. Struts 2 [DB/OL] https://baike.baidu.com/item/Struts%202/2187934?fr=aladdin

[5] 百度百科. Hibernate [DB/OL] https://baike.baidu.com/item/Hibernate/206989?fr=aladdin

[6] 百度百科. DWR [DB/OL] https://baike.baidu.com/item/DWR/1329855?fr=aladdin

[7] 王珊, 萨师煊. 数据库系统概论.第 5 版[M]. 高等教育出版社, 2014.

[8] 软件需求工程 康雁 科学出版社，2011

[9] 魏兴国. HTTP 和 HTTPS 协议安全性分析[J]. 程序员, 2007(7):53-55.

[10] 阿里巴巴. 阿里巴巴 Java 开发手册[M]. 阿里巴巴集团. 2016

致谢

我首先要感谢我的论文指导老师邓广慧。邓老师对我论文给出了许多指导性的意见，在论文撰写过程中及时对我遇到的困难和疑惑给予悉心指点，提出了许多有益的改善性意见，同时，还要感谢我的同学还有室友们，我们互相学习，互相帮忙，这才使得本项目能顺利完成。

此外，还要感谢朋友以及同学们在论文编写中带给的大力支持和帮忙，给我带来极大的启发。也要感谢参考文献中的作者们，透过他们的研究文章，使我对研究课题有了很好的出发点。

最后，谢谢论文评阅老师们的辛苦工作。衷心感谢我的家人、朋友，以及同学们，真是在他们的鼓励和支持下我才得以顺利完成此论文。

附录 A 部分源代码

applicationContexe.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">


    <!-- data connection -->
    <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-
method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
        <property name="url"
value="jdbc:mysql://localhost:3306/yinyousentence?useUnicode=true&amp;characterEncoding=
UTF-
8&amp;useJDBCCompliantTimezoneShift=true&amp;useLegacyDatetimeCode=false&amp;serve
rTimezone=Hongkong"></property>
        <property name="username" value="root"/>
        <property name="password" value=""/>
        <!--<property name="password" value="TsingHua_981127"/>-->
    </bean>


    <!-- session factory -->
    <bean id="sessionFactory"
class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
                <prop key="hibernate.show_sql">true</prop>
                <prop
key="hibernate.connection.url">jdbc:mysql://localhost:3306/yinyousentence</prop>
                <prop key="hibernate.connection.driver_class">com.mysql.jdbc.Driver</prop>
            </props>
        </property>
        <property name="mappingLocations">
            <list>
```

```
                <value>classpath:Login.hbm.xml</value>
                <value>classpath:UserInfo.hbm.xml</value>
                <value>classpath:CommentReply.hbm.xml</value>
                <value>classpath:GiantInfo.hbm.xml</value>
                <value>classpath:GiantLove.hbm.xml</value>
                <value>classpath:OriginInfo.hbm.xml</value>
                <value>classpath:OriginLove.hbm.xml</value>
                <value>classpath:Sentence.hbm.xml</value>
                <value>classpath:SentenceCollection.hbm.xml</value>
                <value>classpath:SentenceComment.hbm.xml</value>
                <value>classpath:SentenceLove.hbm.xml</value>
                <value>classpath:Tag.hbm.xml</value>
                <value>classpath:TagQuote.hbm.xml</value>
                <value>classpath:UserMessage.hbm.xml</value>
                <value>classpath:Category.hbm.xml</value>
                <value>classpath:SentenceCategory.hbm.xml</value>
                <value>classpath:CollectionSentence.hbm.xml</value>
                <value>classpath:UserFollow.hbm.xml</value>
                <value>classpath:CommentReply.hbm.xml</value>
                <value>classpath:CollectionLove.hbm.xml</value>
        </list>
</property>
<property name="annotatedClasses">
        <list>
                <value>bean.Login</value>
                <value>bean.UserInfo</value>
                <value>bean.CommentReply</value>
                <value>bean.GiantInfo</value>
                <value>bean.GiantLove</value>
                <value>bean.OriginInfo</value>
                <value>bean.OriginLove</value>
                <value>bean.Sentence</value>
                <value>bean.SentenceCollection</value>
                <value>bean.SentenceComment</value>
                <value>bean.SentenceLove</value>
                <value>bean.Tag</value>
                <value>bean.TagQuote</value>
                <value>bean.UserMessage</value>
                <value>bean.Category</value>
                <value>bean.SentenceCategory</value>
                <value>bean.CollectionSentence</value>
                <value>bean.UserFollow</value>
                <value>bean.CollectionLove</value>
        </list>
```

```
        </property>
    </bean>

    <bean id="loginRegisterDao" class="dao.LoginRegisterDao">
        <property name="sessionFactory" ref="sessionFactory"></property>
    </bean>

    <bean id="dwrLogin" class="service.LoginService">
        <property name="loginRegisterDao" ref="loginRegisterDao"></property>
    </bean>

    <bean id="dwrRegister" class="service.RegisterService">
        <property name="loginRegisterDao" ref="loginRegisterDao"></property>
    </bean>

    <bean id="dateUtil" class="util.DateUtil"/>

    <bean id="dwrEditInfo" class="service.EditUserInfoService">
    </bean>

    <bean id="informationDao" class="dao.InformationDao">
        <property name="sessionFactory" ref="sessionFactory"></property>
    </bean>

    <bean id="publishSentenceEntity" class="pageEntity.PublishSentenceEntity">
        <property name="informationDao" ref="informationDao"></property>
    </bean>

    <bean id="publishSentenceDao" class="dao.PublishSentenceDao">
        <property name="sessionFactory" ref="sessionFactory"></property>
    </bean>

    <bean id="peopleDao" class="dao.PeopleDao">
        <property name="sessionFactory" ref="sessionFactory"></property>
    </bean>

    <bean id="peopleEntity" class="pageEntity.PeopleEntity">
        <property name="peopleDao" ref="peopleDao"></property>
    </bean>

    <bean id="sentenceDao" class="dao.SentenceDao">
        <property name="sessionFactory" ref="sessionFactory"></property>
    </bean>
```

```
<bean id="sentenceEntity" class="pageEntity.SentenceEntity">
    <property name="sentenceDao" ref="sentenceDao"></property>
</bean>

<bean id="dwrSentenceInfo" class="service.SentenceInfoService">
    <property name="sentenceDao" ref="sentenceDao"/>
    <property name="peopleDao" ref="peopleDao"/>
    <property name="collectionDao" ref="collectionDao"/>
</bean>

<bean id="commentDao" class="dao.CommentDao">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>

<bean id="commentEntity" class="pageEntity.CommentEntity">
    <property name="commentDao" ref="commentDao"/>
</bean>

<bean id="dwrLoginCheck" class="service.LoginCheckService"/>

<bean id="dwrCollect" class="service.CollectService">
    <property name="collectionDao" ref="collectionDao"/>
</bean>

<bean id="collectionDao" class="dao.CollectionDao">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>

<bean id="collectionEntity" class="pageEntity.CollectionEntity">
    <property name="collectionDao" ref="collectionDao"/>
</bean>

<bean id="giantDao" class="dao.GiantDao">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>

<bean id="indexEntity" class="pageEntity.IndexEntity">
    <property name="collectionDao" ref="collectionDao"/>
    <property name="informationDao" ref="informationDao"/>
    <property name="giantDao" ref="giantDao"/>
    <property name="sentenceDao" ref="sentenceDao"/>
</bean>

<bean id="dwrPeople" class="service.PeopleService">
```

```
        <property name="peopleDao" ref="peopleDao"/>
    </bean>

    <bean id="giantInfoEntity" class="pageEntity.GiantInfoEntity">
        <property name="giantDao" ref="giantDao"/>
        <property name="peopleDao" ref="peopleDao"/>
    </bean>

    <bean id="dwrGiant" class="service.GiantService">
        <property name="giantDao" ref="giantDao"/>
        <property name="publishSentenceDao" ref="publishSentenceDao"/>
        <property name="originDao" ref="originDao"/>
        <property name="collectionDao" ref="collectionDao"/>
    </bean>

    <bean id="originDao" class="dao.OriginDao">
        <property name="sessionFactory" ref="sessionFactory"/>
    </bean>

    <bean id="originInfoEntity" class="pageEntity.OriginInfoEntity">
        <property name="originDao" ref="originDao"/>
        <property name="giantDao" ref="giantDao"/>
        <property name="peopleDao" ref="peopleDao"/>
    </bean>



</beans>
```

struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
        "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
    <!-- struts 的 action 配置文件  -->
    <!--  将 action 托管给 spring -->
    <constant name="struts.objectFactory" value="spring" />

    <!--  所有的 action 都应该放在对应的 package 下  -->
```

```
    <!--action 的 class 属性不再是具体的类,而是 spring 配置文件配置的 bean id-
->
    <package name="yinyousentence" namespace="/" extends="struts-default">

        <default-action-ref name="index"></default-action-ref>

        <!--进入登录注册页面的 action-->
        <action name="toLoginOrRegister"
class="action.ToLoginOrRegisterAction" method="toLoginOrRegister">
            <result name="success">/WEB-INF/View/login.jsp</result>
        </action>

        <action name="login" class="action.LoginAction" method="doLogin">
            <result name="success" type="redirectAction">index</result>
            <result name="input"
type="redirectAction">toLoginOrRegister</result>
        </action>

        <action name="register" class="action.RegisterAction"
method="doRegister">
            <result name="success" type="redirectAction">toEditInfo</result>
            <result name="input">toLoginOrRegister.action</result>
            <result name="error">toLoginOrRegister.action</result>
        </action>

        <action name="toEditInfo" class="action.ToEditInfoAction"
method="toEditInfoAction">
            <result name="success">/WEB-INF/View/edit.jsp</result>
            <result name="input"
type="redirectAction">toLoginOrRegister</result>
        </action>

        <action name="edit" class="action.EditAction" method="doEdit">
            <result name="success" type="redirectAction">index</result>
            <result name="input">/WEB-INF/View/index.jsp</result>
            <result name="error">/WEB-INF/View/index.jsp</result>
            <interceptor-ref name="fileUpload">
                <param name="maximumSize">5242880</param>
                <param
name="allowedTypes">image/jpeg,image/gif,image/jpg,image/png</param>
            </interceptor-ref>
            <interceptor-ref name="defaultStack"></interceptor-ref>
        </action>
```

```
        <action name="index" class="action.ToIndexAction" method="execute">
            <result name="success">/WEB-INF/View/index.jsp</result>
            <result name="input">/WEB-INF/View/index.jsp</result>
        </action>

        <action name="publishSentence" class="action.ToPublishSentenceAction"
method="execute">
            <result name="success">/WEB-INF/View/publishSentence.jsp</result>
            <result name="input">/WEB-INF/View/publishSentence.jsp</result>
        </action>

        <action name="doPublishSentence" class="action.PublishSentenceAction"
method="execute">
            <result name="success" type="redirectAction">
                <param name="actionName">sentence</param>
                <param name="sentenceId">${sentenceId}</param>
            </result>
            <result name="input">/WEB-INF/View/sentenceInfo.jsp</result>
            <result name="error">/WEB-INF/View/sentenceInfo.jsp</result>
        </action>

        <action name="toPeople" class="action.ToPeopleAction" method="execute">
            <result name="success">/WEB-INF/View/people.jsp</result>
            <result name="input">/WEB-INF/View/people.jsp</result>
            <result name="error">/WEB-INF/View/people.jsp</result>
        </action>

        <action name="logout" class="action.LogoutAction" method="execute">
            <result name="success"
type="redirectAction">toLoginOrRegister</result>
        </action>

        <action name="giant" class="action.ToGiantAction" method="execute">
            <result name="success">/WEB-INF/View/giantInfo.jsp</result>
        </action>

        <action name="origin" class="action.ToOriginAction" method="execute">
            <result name="success">/WEB-INF/View/origin.jsp</result>
        </action>

        <action name="sentence" class="action.ToSentenceInfoAction"
method="execute">
            <result name="success">/WEB-INF/View/sentenceInfo.jsp</result>
```

```
        </action>
        <action name="userLove" class="action.ToUserLoveAction"
method="execute">
            <result name="success">/WEB-INF/View/userLove.jsp</result>
        </action>
        <action name="giantSentence" class="action.ToGiantSentenceAction"
method="execute">
            <result name="success">/WEB-INF/View/giantSentence.jsp</result>
        </action>

        <action name="comment" class="action.CommentAction"
method="execute">
            <result name="success" type="redirectAction">
                <param name="actionName">sentence</param>
                <param name="sentenceId">${sentenceId}</param>
            </result>
            <result name="input" type="redirectAction">
                <param name="actionName">sentence</param>
                <param name="sentenceId">${sentenceId}</param>
            </result>
            <result name="error">/WEB-INF/View/sentenceInfo.jsp</result>
        </action>

        <action name="reply" class="action.ReplyAction" method="execute">
            <result name="success" type="redirectAction">
                <param name="actionName">sentence</param>
                <param name="sentenceId">${sentenceId}</param>
            </result>
            <result name="input" type="redirectAction">
                <param name="actionName">sentence</param>
                <param name="sentenceId">${sentenceId}</param>
            </result>
            <result name="error">/WEB-INF/View/sentenceInfo.jsp</result>
        </action>

        <action name="collectList" class="action.ToCollectiListAction">
            <result name="success">/WEB-INF/View/collectList.jsp</result>
        </action>

        <action name="search" class="action.ToSearchAction">
            <result name="success">/WEB-INF/View/search.jsp</result>
        </action>

    </package>
```

```
</struts>
```

SentenceEntity

```
package pageEntity;

import auxiliary.UserAuxiliary;
import bean.*;
import dao.SentenceDao;
import org.directwebremoting.WebContextFactory;
import org.springframework.beans.factory.annotation.Autowired;

import javax.servlet.http.HttpSession;
import java.util.ArrayList;
import java.util.List;

/**
 * @ClassName SentenceEntity
 * @Description TODO
 * @Author hasee
 * @Date 2018-07-08 15:53
 * Version 1.0
 */
public class SentenceEntity {
    @Autowired
    private SentenceDao sentenceDao;

    private Boolean original;


    private Sentence sentence;
    private GiantInfo giantInfo;
    private UserInfo userInfo;
    private List<Sentence> giantSentences = new ArrayList<Sentence>();
    private List<UserAuxiliary> loveUsers = new ArrayList<UserAuxiliary>();
    private List<Sentence> originSentences = new ArrayList<Sentence>();
    private List<Tag> tags = new ArrayList<Tag>();

    private OriginInfo originInfo;
    private Boolean userLove = false;
    private long commentNum = 0;
```

```
    // 专门给 people 页面使用的一个属性
    private Boolean peopleLove;

    // 专门给 people 页面使用的初始化方式
    public void init(long sentenceId, long myId, long uId){
        init(sentenceId,myId);
        long loveReord = sentenceDao.checkUserLoveSentence(uId,sentenceId);
        if(loveReord == 0){
            peopleLove = false;
        }else{
            peopleLove = true;
        }
    }


    // 初始化数据，这个 id 必须判断一定存在才会到这
    public void init(long sentenceId,long myId){
        if(giantSentences != null){
            giantSentences.clear();
        }
        if(loveUsers != null){
            loveUsers.clear();
        }

        sentence = sentenceDao.getSentenceById(sentenceId);

        int org = sentence.getIsOriginal();
        System.out.println("是否原创：" + sentence.getIsOriginal() + " " + org);
        if(org == 0){
            original = false;
        }else{
            original = true;
        }


        if(sentence.getAuthorId() != 0){
            giantInfo = sentenceDao.getGiantInfoById(sentence.getAuthorId());
            giantSentences =
sentenceDao.getSentencesByAuthorId(sentence.getAuthorId());
            originSentences =
sentenceDao.getSentencesByOriginId(sentence.getOriginId());
            if(giantSentences != null && giantSentences.size() > 3){
                giantSentences.subList(0,3);
            }
```

```
            if(originSentences != null && originSentences.size() > 3){
                originSentences.subList(0,3);
            }

        }else{
            giantInfo = null;
            giantSentences = null;
            originSentences = null;
        }

        if(sentence.getOriginId() != 0){
            originInfo = sentenceDao.getOriginInfoById(sentence.getOriginId());
        }else{
            originInfo = null;
        }

        userInfo = sentenceDao.getUserInfoById(sentence.getPublisherId());
        List<Long> userIds =
sentenceDao.getLoveSentenceUserIdBySentenceId(sentence.getId());

        for(Long id : userIds){
            if(id.equals(myId)){
                // System.out.println("屏蔽了一个:" + myId);
                continue;
            }
            //System.out.println("里面至少有一个: " + id);
            UserInfo userInfo2 = new UserInfo();
            userInfo2 = sentenceDao.getUserInfoById(id);
            UserAuxiliary userAuxiliary = new UserAuxiliary();
            userAuxiliary.setUserInfo(userInfo2);

userAuxiliary.setLoveOrNot(sentenceDao.checkIfUserFollowUser(myId,userInfo2.ge
tId()));
            loveUsers.add( userAuxiliary);
        }
        if(loveUsers != null && loveUsers.size() > 3){
            loveUsers = loveUsers.subList(0,3);
        }



        initTags();

        // 判断用户是否喜欢该句子
```

```
        initLoveCheck(sentenceId, myId);

        commentNum = sentenceDao.getCommentNumBySentenceId(sentenceId);
    }

    // 判断用户是否喜欢该句子
    public void initLoveCheck(long sentenceId,long userId){
        if(userId == 0){
            return;
        }
        long love = sentenceDao.checkUserLoveSentence(userId,sentenceId);
        if(love != 0){
            userLove = true;
        }else{
            userLove = false;
        }
    }

    // 初始化标签列表
    public void initTags(){
        if(tags != null){
            tags.clear();
        }
        String s_tag = sentence.getTag();
        String[] array_tags = s_tag.split("\\s+");
        for(String s : array_tags){
            Tag tag = sentenceDao.getTagByName(s);
            if(tag != null){
                tags.add(tag);
            }
        }
    }

    public SentenceDao getSentenceDao() {
        return sentenceDao;
    }

    public void setSentenceDao(SentenceDao sentenceDao) {
        this.sentenceDao = sentenceDao;
    }

    public Sentence getSentence() {
        return sentence;
    }
```

```
public void setSentence(Sentence sentence) {
        this.sentence = sentence;
}

public GiantInfo getGiantInfo() {
        return giantInfo;
}

public void setGiantInfo(GiantInfo giantInfo) {
        this.giantInfo = giantInfo;
}

public List<Sentence> getGiantSentences() {
        return giantSentences;
}

public void setGiantSentences(List<Sentence> giantSentences) {
        this.giantSentences = giantSentences;
}

public UserInfo getUserInfo() {
        return userInfo;
}

public void setUserInfo(UserInfo userInfo) {
        this.userInfo = userInfo;
}



public OriginInfo getOriginInfo() {
        return originInfo;
}

public void setOriginInfo(OriginInfo originInfo) {
        this.originInfo = originInfo;
}

public Boolean getOriginal() {
        return original;
}
```

```
public void setOriginal(Boolean original) {
    this.original = original;
}

public List<Tag> getTags() {
    return tags;
}

public void setTags(List<Tag> tags) {
    this.tags = tags;
}

public List<Sentence> getOriginSentences() {
    return originSentences;
}

public void setOriginSentences(List<Sentence> originSentences) {
    this.originSentences = originSentences;
}


public Boolean getUserLove() {
    return userLove;
}

public void setUserLove(Boolean userLove) {
    this.userLove = userLove;
}

public long getCommentNum() {
    return commentNum;
}

public void setCommentNum(long commentNum) {
    this.commentNum = commentNum;
}

public Boolean getPeopleLove() {
    return peopleLove;
}

public void setPeopleLove(Boolean peopleLove) {
    this.peopleLove = peopleLove;
}
```

```
    public List<UserAuxiliary> getLoveUsers() {
        return loveUsers;
    }

    public void setLoveUsers(List<UserAuxiliary> loveUsers) {
        this.loveUsers = loveUsers;
    }
}
```

CollectService

```
package service;

import auxiliary.CollectionAuxiliary;
import bean.CollectionSentence;
import bean.SentenceCollection;
import bean.UserInfo;
import dao.CollectionDao;
import dwrPOJO.CollectionCollectPOJO;
import org.directwebremoting.WebContextFactory;
import org.springframework.beans.factory.annotation.Autowired;

import javax.servlet.http.HttpSession;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

/**
 * @ClassName CollectService
 * @Description TODO
 * @Author hasee
 * @Date 2018-07-10 18:45
 * Version 1.0
 */
public class CollectService {

    @Autowired
    private CollectionDao collectionDao;
    /**
    * @author hasee
    * @Description  收藏或取消收藏
    * @Date 23:43 2018-07-10
    * @Param [sentenceId,collectionId]
```

```
    * @return auxiliary.CollectionAuxiliary
    **/
    public CollectionCollectPOJO doCollect(String sentenceId, String collectionId){
        CollectionCollectPOJO collectionCollectPOJO = new
CollectionCollectPOJO();
        long stId = 0;
        long ctId = 0;
        try {
            stId = Long.valueOf(sentenceId);
            ctId = Long.valueOf(collectionId);
        }catch (Exception e){
            collectionCollectPOJO.setSuccess(false);
            collectionCollectPOJO.setReason("参数错误，请刷新页面");
            return collectionCollectPOJO;
        }
        // 判断该句子集是否为当前用户发布的，不是就不能添加
        HttpSession session = WebContextFactory.get().getSession();
        UserInfo userInfo = (UserInfo) session.getAttribute("userInfo");
        if(userInfo == null){
            collectionCollectPOJO.setSuccess(false);
            collectionCollectPOJO.setReason("请先登录");
            return collectionCollectPOJO;
        }
        if(! collectionDao.checkCollectionBelongToUser(userInfo.getId(),ctId)){
            collectionCollectPOJO.setSuccess(false);
            collectionCollectPOJO.setReason("参数错误，请刷新页面");
            return collectionCollectPOJO;
        }

        long csId = collectionDao.getCollectionSentenceId(stId,ctId);
        if(csId == 0){
            // 表示还没有收藏，那么就给他收藏
            CollectionSentence collectionSentence = new CollectionSentence();
            collectionSentence.setSentenceId(stId);
            collectionSentence.setCollectionId(ctId);
            collectionDao.addSentenceIntoCollection(collectionSentence);
            collectionCollectPOJO.setCollect(true);
        }else{
            // 已收藏
            collectionDao.removeSentenceFromCollection(csId,ctId);
            collectionCollectPOJO.setCollect(false);
        }
```

```
collectionCollectPOJO.setSentenceNum(collectionDao.getSentenceNumInCollection(
ctId));
            System.out.println(collectionCollectPOJO.getSentenceNum());
            collectionCollectPOJO.setSuccess(true);
            return collectionCollectPOJO;
        }

        public CollectionCollectPOJO addNewCollection(String collectionName){
            CollectionCollectPOJO collectionCollectPOJO = new
CollectionCollectPOJO();

            HttpSession session = WebContextFactory.get().getSession();
            UserInfo userInfo = (UserInfo) session.getAttribute("userInfo");
            if(userInfo == null){
                collectionCollectPOJO.setSuccess(false);
                collectionCollectPOJO.setReason("请先登录");
                return collectionCollectPOJO;
            }

            // 判断当前用户是否已经有一个这个名字的句子集了

if(collectionDao.isColletionNameExistForTheUser(userInfo.getId(),collectionName))
{
                // 如果已经存在
                collectionCollectPOJO.setSuccess(false);
                collectionCollectPOJO.setReason("您已经有一个该名字的句子集啦，
请更换名称后重试哦");
                return collectionCollectPOJO;
            }

            // 新增一个句子集
            SentenceCollection sentenceCollection = new SentenceCollection();
            sentenceCollection.setImgPath(userInfo.getHeadPath());
            sentenceCollection.setIntroduction("");
            sentenceCollection.setLoveNum(0);
            sentenceCollection.setName(collectionName);
            sentenceCollection.setPublishDate(new
Timestamp(System.currentTimeMillis()));
            sentenceCollection.setPublisherId(userInfo.getId());
            sentenceCollection.setSentenceNum(0);
            collectionDao.addNewCollection(sentenceCollection);
            collectionCollectPOJO.setSuccess(true);
            return collectionCollectPOJO;
        }
```

```
    public List<CollectionAuxiliary> initCollectionList(long sentenceId){
        HttpSession session = WebContextFactory.get().getSession();
        UserInfo userInfo = (UserInfo) session.getAttribute("userInfo");
        long userId = userInfo.getId();
        List<CollectionAuxiliary> collectionAuxiliaries = new
ArrayList<CollectionAuxiliary>();
        List<SentenceCollection> sentenceCollections =
collectionDao.getSentenceCollectionsByUserId(userId);
        for(SentenceCollection sc : sentenceCollections){
            CollectionAuxiliary collectionAuxiliary = new CollectionAuxiliary();
            collectionAuxiliary.setSentenceCollection(sc);

collectionAuxiliary.setCollect(collectionDao.isSentenceCollectedInCollection(sentenc
eId,sc.getId()));
            collectionAuxiliaries.add(collectionAuxiliary);
        }
        return collectionAuxiliaries;
    }

    public CollectionDao getCollectionDao() {
        return collectionDao;
    }

    public void setCollectionDao(CollectionDao collectionDao) {
        this.collectionDao = collectionDao;
    }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

    <!-- 在 struts2 之前加一个 dwr -->
    <servlet>
        <servlet-name>dwr-invoker</servlet-name>
```

```
            <servlet-class>org.directwebremoting.servlet.DwrServlet</servlet-class>
            <!-- 是否允许调试，如果要在浏览器中调试则必须设置为 true -->
            <init-param>
                <param-name>debug</param-name>
                <param-value>true</param-value>
            </init-param>
            <!-- 如果允许跨域请求，则必须将此值设置为 false，默认值为 true -->
            <init-param>
                <param-name>crossDomainSessionSecurity</param-name>
                <param-value>true</param-value>
            </init-param>
            <init-param>
                <param-name>allowScriptTagRemoting</param-name>
                <param-value>true</param-value>
            </init-param>
            <init-param>
                <param-name>jsonpEnabled</param-name>
                <param-value>true</param-value>
            </init-param>
        </servlet>
        <servlet-mapping>
            <servlet-name>dwr-invoker</servlet-name>
            <url-pattern>/dwr/*</url-pattern>
        </servlet-mapping>


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



        <!-- 过滤器，解决编码问题 -->
        <filter>
            <filter-name>encoding</filter-name>
            <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
            <init-param>
```

```
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>encoding</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>









    <!--struts2 测试加入-->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>WEB-INF/applicationContext.xml</param-value>
    </context-param>
</web-app>
```

dwr.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dwr PUBLIC "-//GetAhead Limited//DTD Direct Web Remoting
2.0//EN" "http://getahead.ltd.uk/dwr/dwr30.dtd">
<dwr>
    <allow>
        <create creator="spring" javascript="dwrLogin">
            <param name="beanName" value="dwrLogin" />
        </create>
        <create creator="spring" javascript="dwrRegister">
            <param name="beanName" value="dwrRegister" />
        </create>
        <create creator="spring" javascript="dwrEditInfo">
            <param name="beanName" value="dwrEditInfo"/>
        </create>
```

```
        <create creator="spring" javascript="dwrSentenceInfo">
            <param name="beanName" value="dwrSentenceInfo"/>
        </create>

        <create creator="spring" javascript="dwrLoginCheck">
            <param name="beanName" value="dwrLoginCheck"/>
        </create>

        <create creator="spring" javascript="dwrCollect">
            <param name="beanName" value="dwrCollect"/>
        </create>

        <create creator="spring" javascript="dwrPeople">
            <param name="beanName" value="dwrPeople"/>
        </create>

        <create creator="spring" javascript="dwrGiant">
            <param name="beanName" value="dwrGiant"/>
        </create>

        <convert converter="bean" match="dwrPOJO.*"/>
        <convert converter="bean" match="bean.*"/>
        <convert converter="bean" match="pageEntity.*"/>
        <convert converter="bean" match="auxiliary.CollectionAuxiliary"/>
        <convert converter="bean" match="java.lang.StackTraceElement" />
        <convert converter="exception" match="java.lang.Exception"/>
    </allow>
</dwr>
```

PublishSentenceAction

```
package action;

import bean.*;
import com.opensymphony.xwork2.ActionSupport;
import dao.PublishSentenceDao;
import org.apache.struts2.interceptor.SessionAware;
import org.springframework.beans.factory.annotation.Autowired;

import java.sql.Date;
import java.sql.Timestamp;
import java.util.Map;
```

```
/**
 * @ClassName PublishSentenceAction
 * @Description TODO
 * @Author hasee
 * @Date 2018-07-07 19:30
 * Version 1.0
 */
public class PublishSentenceAction extends ActionSupport implements SessionAware{
    @Autowired
    private PublishSentenceDao publishSentenceDao;

    private Map session;

    private String collectSelect;
    private String sentenceContent;
    private String original;
    private String authorName;
    private String bookName;
    private String categorySelect;
    private String tagInput;

    private long sentenceId;


    @Override
    public String execute() throws Exception {
        UserInfo userInfo = (UserInfo) session.get("userInfo");
        if(userInfo == null){
            return ERROR;
        }

        Sentence sentence = new Sentence();
        // 1、将该句子存入数据库
        sentence.setContent(sentenceContent);
        sentence.setPublisherId(userInfo.getId());
        if(original == null){
            sentence.setIsOriginal((byte) 0);
        }else{
            sentence.setIsOriginal((byte) 1);
        }
        long authorId = 0;
        if(!"".equals(authorName.trim())){
            authorId = getAuthorId(authorName);
```

```
            }
            long originId = 0;
            if(!"".equals(bookName.trim())){
                originId = getOriginId(authorId,bookName);
            }
            Timestamp publishTime = new Timestamp(System.currentTimeMillis());
            // 存储句子
            sentence.setAuthorId(authorId);
            sentence.setOriginId(originId);
            sentence.setPublishTime(publishTime);
            sentence.setLoveNum(0);
            sentence.setTag(tagInput);
            publishSentenceDao.addSentence(sentence);


            // 处理一下标签
            initTags(sentence.getId(),tagInput);

            // 如果选择中了句子集，将句子加入到句子集
            if(! "0".equals(collectSelect.trim())){

addSentenceToSentenceCollection(sentence.getId(),collectSelect.trim());
            }

            // 添加句子到对应的类别
            if(original != null){
                // 是原创句子
                long categorySelectId = Long.valueOf(categorySelect);
                if(categorySelectId != (long) 15){
                    // 如果是原创但是用户选择的分类不是原创，那么既要存到原
创分类，也要存到对应的分类
                    addSentenceIntoCategory(sentence.getId(),(long) 15);
                }
                addSentenceIntoCategory(sentence.getId(),categorySelectId);
            }else{
                // 非原创
                long categorySelectId = Long.valueOf(categorySelect);
                addSentenceIntoCategory(sentence.getId(),categorySelectId);
            }

            this.sentenceId = sentence.getId();

            return SUCCESS;
        }
```

```
/**
* @author hasee
* @Description  获得作者 ID
* @Date 22:28 2018-07-07
* @Param [userName]
* @return long
**/
public long getAuthorId(String authorName){
    // 判断作者是否存在，不存在就新建一个
    if(! publishSentenceDao.isAuthorExist(authorName)){
        GiantInfo giantInfo = new GiantInfo();
        giantInfo.setImgPath("/imgs/sys/defaultGiant.jpg");
        giantInfo.setIntroduction("");
        giantInfo.setLoveNum(0);
        giantInfo.setName(authorName);
        publishSentenceDao.addGiant(giantInfo);
    }
    long id = publishSentenceDao.getAuthorId(authorName);
    return id;
}

// 获得出处 ID
public long getOriginId(long authorId, String originName){
    // 判断作者是否存在，不存在就新建一个
    if(! publishSentenceDao.isOriginExist(originName)){
        OriginInfo originInfo = new OriginInfo();
        originInfo.setAuthorId(authorId);
        originInfo.setImgPath("/imgs/sys/defaultOrigin.png");
        originInfo.setIntroduction("");
        originInfo.setLoveNum(0);
        if (!originName.startsWith("《")){
            originName = "《" + originName;
        }
        if(! originName.endsWith("》")){
            originName = originName + "》";
        }
        originInfo.setName(originName);

        publishSentenceDao.addOrigin(originInfo);
    }
    long id = publishSentenceDao.getOriginId(originName);
    return id;
```

```
}

// 处理下标签
public void initTags(long sentenceId, String tagInput){
    tagInput = tagInput.trim();
    String[] tags = tagInput.split("\\s+");
    if(tags != null && tags.length > 0){
        for(String tag : tags){
            // 依次将标签存入数据库，如果该标签不存在，则创建一个
            long tagId = 0;
            if(! publishSentenceDao.isTagExist(tag)){
                Tag tag1 = new Tag();
                if(tag.length() >= 25){
                    tag =tag.substring(0,24);
                }
                tag1.setName(tag);
                tag1.setQuoteNum(1);
                publishSentenceDao.addTag(tag1);
                tagId = tag1.getId();
                System.out.println("新：  " + tagId);
            }else{
                tagId = publishSentenceDao.getTagId(tag);
                System.out.println("旧：  " + tagId);
                publishSentenceDao.addTagQuote(tagId);
            }
            // 对标签引用表进操作
            TagQuote tagQuote = new TagQuote();
            tagQuote.setSentenceId(sentenceId);
            tagQuote.setTagId(tagId);
            publishSentenceDao.setTagQuote(tagQuote);
        }
    }
}

// 将句子加入句子集
public void addSentenceToSentenceCollection(long sentenceId, String clt_id){
    long collectionId = 0;
    try{
        collectionId = Long.valueOf(clt_id);
    }catch (Exception e){
        e.printStackTrace();
        return;
    }
    if(collectionId == 0){
```

```
            return;
        }
        CollectionSentence collectionSentence = new CollectionSentence();
        collectionSentence.setCollectionId(collectionId);
        collectionSentence.setSentenceId(sentenceId);
        publishSentenceDao.putSentenceIntoCollection(collectionSentence);
}

// 将句子存入对应分类表
public void addSentenceIntoCategory(long sentenceId, long categoryId){
        SentenceCategory sentenceCategory = new SentenceCategory();
        sentenceCategory.setSentenceId(sentenceId);
        sentenceCategory.setCategoryId(categoryId);
        publishSentenceDao.putSentenceIntoCategory(sentenceCategory);
}

public String getCollectSelect() {
        return collectSelect;
}

public void setCollectSelect(String collectSelect) {
        this.collectSelect = collectSelect;
}

public String getSentenceContent() {
        return sentenceContent;
}

public void setSentenceContent(String sentenceContent) {
        this.sentenceContent = sentenceContent;
}

public String getOriginal() {
        return original;
}

public void setOriginal(String original) {
        this.original = original;
}

public String getAuthorName() {
        return authorName;
}
```

```
public void setAuthorName(String authorName) {
    this.authorName = authorName;
}

public String getBookName() {
    return bookName;
}

public void setBookName(String bookName) {
    this.bookName = bookName;
}

public String getCategorySelect() {
    return categorySelect;
}

public void setCategorySelect(String categorySelect) {
    this.categorySelect = categorySelect;
}

public String getTagInput() {
    return tagInput;
}

public void setTagInput(String tagInput) {
    this.tagInput = tagInput;
}

@Override
public void setSession(Map<String, Object> session) {
    this.session = session;
}


public void setPublishSentenceDao(PublishSentenceDao publishSentenceDao) {
    this.publishSentenceDao = publishSentenceDao;
}


public long getSentenceId() {
    return sentenceId;
}

public void setSentenceId(long sentenceId) {
```

```
            this.sentenceId = sentenceId;
        }
}
```

ToUserLoveAction

```
package action;

import bean.*;
import com.opensymphony.xwork2.ActionSupport;
import dao.*;
import org.apache.struts2.interceptor.SessionAware;
import org.springframework.beans.factory.annotation.Autowired;
import pageEntity.PeopleEntity;
import pageEntity.SentenceEntity;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

/**
 * @ClassName ToUserLoveAction
 * @Description TODO
 * @Author hasee
 * @Date 2018-07-08 23:22
 * Version 1.0
 */
public class ToUserLoveAction extends ActionSupport implements SessionAware{

    // 种类，0:句子  1：  用户
    private String type;

    // 种类内部 Id:
    // 对句子：0：推荐句子，1：热门句子，2：热门原创，3：最新发布，4：分类句子，5：标签句子
    // 对用户：0：喜欢句子，1：喜欢名家，2：喜欢作品。3：关注某一用户，4：某用户的关注
    private String typeId;

    // 句子内部种类细分后，所需要的一些参数（并非所有 typeId 都会需要一个 contentId）
    private String contentId;
```

```
    private Map session;

    @Autowired
    private SentenceDao sentenceDao;

    @Autowired
    private PeopleDao peopleDao;

    @Autowired
    private InformationDao informationDao;

    @Autowired
    private GiantDao giantDao;

    @Autowired
    private OriginDao originDao;


    // 显示的分类: 0：推荐句子，1：热门句子，2：热门原创，3：最新发布，
4：分类句子，5：标签句子
    //6：喜欢某句子用户，7：喜欢某名家用户，8：喜欢某出处用户，9：某用
户关注的用户，10：某用户的粉丝
    private long showCategory;
    // 显示在页面上的标题（分为句子或用户）
    private String showContent;
    private String showUrl;
    // 显示的列表
    private List<SentenceEntity> sentenceEntities;
    private List<PeopleEntity> peopleEntities;



    @Override
    public String execute() throws Exception {

        System.out.println(type + "    " + typeId + "    " + contentId);
        long myType = 0;
        long myTypeId = 0;
        long myContentId = 0;
        try {
            myType = Long.valueOf(type);
            myTypeId = Long.valueOf(typeId);
```

```
            if(myTypeId >= 4 ){
                myContentId = Long.valueOf(contentId);
            }
        }catch (Exception e){
            return ERROR;
        }
        if(myType == 0){
            if(handleSentences()){

            }else{
                return ERROR;
            }
        }else if(myType == 1){
            if(handleUsers()){

            }else{
                return ERROR;
            }
        }else{
            return ERROR;
        }

        session.put("showCategory",showCategory);
        session.put("showContent",showContent);
        session.put("showUrl",showUrl);
        session.put("sentenceEntities",sentenceEntities);
        session.put("peopleEntities",peopleEntities);



        return SUCCESS;
    }

    /**
    * @author hasee
    * @Description  处理句子部分
    * @Date 10:17 2018-07-15
    * @Param []
    * @return void
    **/
    public boolean handleSentences(){
        long myTypeId = 0;
        try {
            myTypeId = Long.valueOf(typeId);
```

```
            }catch (Exception e){
                return false;
            }
            if(myTypeId == 0){
                showCategory = 0;
                initRecommendSentences();
            }else if(myTypeId == 1){
                showCategory = 1;
                initHotSentences();
            }else if(myTypeId == 2){
                showCategory = 2;
                initHotOriginals();
            }else if(myTypeId == 3){
                showCategory = 3;
                initNewPublishs();
            } else{
                long myContentId = 0;
                try{
                    myContentId = Long.valueOf(contentId);
                }catch (Exception e){
                    return false;
                }
                if(myTypeId == 4){
                    showCategory = 4;
                    Category                    category                    =
informationDao.getCategoryByCategoryId(myContentId);
                    if(category == null){
                        return false;
                    }
                    showContent = category.getCategoryName();
                    showUrl = category.getId() + "";
                    if(initCategorySentences(myContentId)){

                    }else {
                        return false;
                    }
                }else if(myTypeId == 5){
                    showCategory = 5;
                    Tag tag = informationDao.getTagByTagId(myContentId);
                    if(tag == null){
                        return false;
                    }
                    showContent = tag.getName();
                    showUrl = tag.getId() + "";
```

```
                if(initTagSentences(myContentId)){

                }else{
                    return false;
                }
            }
        }
        return true;
    }


    /**
    * @author hasee
    * @Description  处理用户部分
    * @Date 10:18 2018-07-15
    * @Param []
    * @return void
    **/
    public boolean handleUsers(){
        long myTypeId = 0;
        try {
            myTypeId = Long.valueOf(typeId);
        }catch (Exception e){
            return false;
        }
        long myContentId = 0;
        try{
            myContentId = Long.valueOf(contentId);
        }catch (Exception e){
            return false;
        }
        if(myTypeId == 0){
            showCategory = 6;

            Sentence sentence = sentenceDao.getSentenceById(myContentId);
            if(sentence == null){
                return false;
            }
            showContent = sentence.getContent();
            showUrl = sentence.getId() + "";

            initLoveSentenceUsers(myContentId);
        }else if(myTypeId == 1){
            showCategory = 7;
```

```
        GiantInfo giant = giantDao.getGiantInfoById(myContentId);
        if(giant == null){
            return false;
        }
        showContent = giant.getName();
        showUrl = giant.getId() + "";

        initLoveGiantUsers(myContentId);
    }else if(myTypeId == 2){
        showCategory = 8;

        OriginInfo originInfo = originDao.getOriginInfoById(myContentId);
        if(originInfo == null){
            return false;
        }
        showContent = originInfo.getName();
        showUrl = originInfo.getId() + "";

        initLoveOriginUsers(myContentId);
    }else if(myTypeId == 3){
        showCategory = 9;

        UserInfo userInfo = peopleDao.getUserInfoById(myContentId);
        if(userInfo == null){
            return false;
        }
        showContent = userInfo.getUserName();
        showUrl = userInfo.getId() + "";

        initFollowUsers(myContentId);
    }else if(myTypeId == 4){
        showCategory = 10;

        UserInfo userInfo = peopleDao.getUserInfoById(myContentId);
        if(userInfo == null){
            return false;
        }
        showContent = userInfo.getUserName();
        showUrl = userInfo.getId() + "";

        initFanUsers(myContentId);
    }else {
        return false;
```

```
        }
        return true;
    }



        /**
    * @author hasee
    * @Description  将句子列表转化为 sentenceEntity 并存入 eneities 中
    * @Date 12:45 2018-07-15
    * @Param [sentences]
    * @return
    **/
    public void putSentencesIntoEntities(List<Sentence> sentences){
        if(sentences == null){
            return;
        }
        long myId = 0;
        UserInfo myInfo = (UserInfo) session.get("userInfo");
        if(myInfo != null){
            myId = myInfo.getId();
        }
        if(sentenceEntities != null){
            sentenceEntities.clear();
        }else {
            sentenceEntities = new ArrayList<SentenceEntity>();
        }
        for(Sentence s: sentences){
            SentenceEntity sentenceEntity = new SentenceEntity();
            sentenceEntity.setSentenceDao(sentenceDao);
            sentenceEntity.init(s.getId(),myId);
            sentenceEntities.add(sentenceEntity);
        }
    }

    /**
    * @author hasee
    * @Description  重载 putSentencesInfoEntities 方法，这个是直接传 id
    * @Date 12:54 2018-07-15
    * @Param [sentenceIds]
    * @return void
    **/
    public void putSentencesInfoEntities(List<Long> sentenceIds){
```

```
        if(sentenceIds == null){
            return;
        }
        long myId = 0;
        UserInfo myInfo = (UserInfo) session.get("userInfo");
        if(myInfo != null){
            myId = myInfo.getId();
        }
        if(sentenceEntities != null){
            sentenceEntities.clear();
        }else {
            sentenceEntities = new ArrayList<SentenceEntity>();
        }
        for(Long s: sentenceIds){
            SentenceEntity sentenceEntity = new SentenceEntity();
            sentenceEntity.setSentenceDao(sentenceDao);
            sentenceEntity.init(s,myId);
            sentenceEntities.add(sentenceEntity);
        }
}



/**
* @author hasee
* @Description  初始化推荐句子
* @Date 12:42 2018-07-15
* @Param []
* @return void
**/
public void initRecommendSentences(){
        List<Sentence> sentences = sentenceDao.getRecommendSentences(30);
        putSentencesIntoEntities(sentences);
}

/**
* @author hasee
* @Description  初始化热门句子
* @Date 12:50 2018-07-15
* @Param []
* @return void
**/
public void initHotSentences(){
        List<Sentence> sentences = sentenceDao.getHotSentences(30);
```

```
        putSentencesIntoEntities(sentences);
    }

    /**
    * @author hasee
    * @Description  初始化热门原创句子
    * @Date 12:51 2018-07-15
    * @Param []
    * @return void
    **/
    public void initHotOriginals(){
        List<Sentence> sentences = sentenceDao.getHotOriginSentences(30);
        putSentencesIntoEntities(sentences);
    }

    /**
    * @author hasee
    * @Description  初始化最新发布句子
    * @Date 12:51 2018-07-15
    * @Param []
    * @return void
    **/
    public void initNewPublishs(){
        List<Sentence> sentences = sentenceDao.getNewPublishSentence(30);
        putSentencesIntoEntities(sentences);
    }



    /**
    * @author hasee
    * @Description  初始化分类下句子
    * @Date 13:12 2018-07-15
    * @Param [category]
    * @return java.lang.Boolean
    **/
    public Boolean initCategorySentences(long categoryId){
        List<Long>                                  sentences                                  =
sentenceDao.getSentenceIdsByCategoryId(categoryId,0,30);
        putSentencesInfoEntities(sentences);
        return true;
    }
```

```
/**
* @author hasee
* @Description  初始化标签下句子
* @Date 13:17 2018-07-15
* @Param [tagId]
* @return java.lang.Boolean
**/
public Boolean initTagSentences(long tagId){
    List<Long> sentences = sentenceDao.getSentenceIdsByTagId(tagId,0,30);
    putSentencesInfoEntities(sentences);
    return true;
}



public void putUserIntoEntities(List<Long> userIds){
    //System.out.println("putUSerIntoEntities");
    //System.out.println(userIds);
    if (userIds == null) {
        return;
    }
    if(peopleEntities != null){
        peopleEntities.clear();
    } else {
        peopleEntities = new ArrayList<PeopleEntity>();
    }
    long myId = 0;
    UserInfo myInfo = (UserInfo) session.get("userInfo");
    if(myInfo != null){
        myId = myInfo.getId();
    }
    for(Long userId : userIds){
        PeopleEntity peopleEntity = new PeopleEntity();
        peopleEntity.setPeopleDao(peopleDao);
        peopleEntity.init(myId, userId);
        peopleEntities.add(peopleEntity);
    }
   // System.out.println(peopleEntities);
}



/**
* @author hasee
```

```
* @Description  初始化喜欢某句子的用户列表
* @Date 13:35 2018-07-15
* @Param [sentenceId]
* @return void
**/
public void initLoveSentenceUsers(long sentenceId){
    List<Long> userIds = peopleDao.getUserIdsBySentenceId(sentenceId,0,30);
    System.out.println(userIds);
    putUserIntoEntities(userIds);
}

/**
* @author hasee
* @Description  根据名家 id 获取喜欢该名家的用户的信息
* @Date 14:09 2018-07-15
* @Param [giantId]
* @return void
**/
public void initLoveGiantUsers(long giantId){
    List<Long> userIds = peopleDao.getUserIdsByGiantId(giantId,0,30);
    putUserIntoEntities(userIds);
}

/**
* @author hasee
* @Description  根据出处 id 获取喜欢这个出处的用户的信息
* @Date 14:09 2018-07-15
* @Param [originId]
* @return void
**/
public void initLoveOriginUsers(long originId){
    List<Long> userIds = peopleDao.getUserIdsByOriginId(originId,0,30);
    putUserIntoEntities(userIds);
}

/**
* @author hasee
* @Description  根据用户 id 获取其所关注的用户的 id
* @Date 14:09 2018-07-15
* @Param [userId]
* @return void
**/
public void initFollowUsers(long userId){
    List<Long> userIds = peopleDao.getUserIdsByFollowerId(userId,0,30);
```

```
        putUserIntoEntities(userIds);
    }

    /**
    * @author hasee
    * @Description  根据用户 Id 获取该用户的粉丝的 id
    * @Date 14:08 2018-07-15
    * @Param [userId]
    * @return void
    **/
    public void initFanUsers(long userId){
        List<Long> userIds = peopleDao.getUserIdsByFollowingId(userId,0,30);
        putUserIntoEntities(userIds);
    }



    @Override
    public void setSession(Map<String, Object> session) {
        this.session = session;
    }


    public List<SentenceEntity> getSentenceEntities() {
        return sentenceEntities;
    }

    public void setSentenceEntities(List<SentenceEntity> sentenceEntities) {
        this.sentenceEntities = sentenceEntities;
    }

    public List<PeopleEntity> getPeopleEntities() {
        return peopleEntities;
    }

    public void setPeopleEntities(List<PeopleEntity> peopleEntities) {
        this.peopleEntities = peopleEntities;
    }

    public SentenceDao getSentenceDao() {
        return sentenceDao;
    }

    public void setSentenceDao(SentenceDao sentenceDao) {
```

```
        this.sentenceDao = sentenceDao;
    }

    public PeopleDao getPeopleDao() {
        return peopleDao;
    }

    public void setPeopleDao(PeopleDao peopleDao) {
        this.peopleDao = peopleDao;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getTypeId() {
        return typeId;
    }

    public void setTypeId(String typeId) {
        this.typeId = typeId;
    }

    public String getContentId() {
        return contentId;
    }

    public void setContentId(String contentId) {
        this.contentId = contentId;
    }

    public Map getSession() {
        return session;
    }


    public long getShowCategory() {
        return showCategory;
    }
```

```
public void setShowCategory(long showCategory) {
    this.showCategory = showCategory;
}

public String getShowContent() {
    return showContent;
}

public void setShowContent(String showContent) {
    this.showContent = showContent;
}

public String getShowUrl() {
    return showUrl;
}

public void setShowUrl(String showUrl) {
    this.showUrl = showUrl;
}

public InformationDao getInformationDao() {
    return informationDao;
}

public void setInformationDao(InformationDao informationDao) {
    this.informationDao = informationDao;
}

public GiantDao getGiantDao() {
    return giantDao;
}

public void setGiantDao(GiantDao giantDao) {
    this.giantDao = giantDao;
}

public OriginDao getOriginDao() {
    return originDao;
}

public void setOriginDao(OriginDao originDao) {
    this.originDao = originDao;
}
```

```
}
```

ReplyAction

```
package action;

import bean.CommentReply;
import bean.SentenceComment;
import bean.UserInfo;
import com.opensymphony.xwork2.ActionSupport;
import dao.CommentDao;
import org.apache.struts2.interceptor.SessionAware;
import org.springframework.beans.factory.annotation.Autowired;

import java.sql.Timestamp;
import java.util.Map;

/**
 * @ClassName ReplyAction
 * @Description TODO
 * @Author hasee
 * @Date 2018-07-10 10:00
 * Version 1.0
 */
public class ReplyAction extends ActionSupport implements SessionAware{

    @Autowired
    private CommentDao commentDao;
    private String type;
    private String commentId;
    private String rpObjId;
    private String content;
    private String sentenceId;
    private Map session;

    @Override
    public String execute() throws Exception {
        // 数据不合法，直接返回
        if(! dataValidationCheck()){
            return INPUT;
        }
        CommentReply commentReply = new CommentReply();
```

```
            // 回复评论
            if("0".equals(type)){
                replyComment(commentReply);
            }else if("1".equals(type)){
                // 回复回复
                replyReply(commentReply);
            }
            SentenceComment sentenceComment =
commentDao.getCommentInfoById(commentReply.getCommentId());
            sentenceId = sentenceComment.getSentenceId() + "";
            return SUCCESS;
        }


        /**
        * @author hasee
        * @Description 回复评论
        * @Date 16:01 2018-07-10
        * @Param [commentReply]
        * @return void
        **/
        public void replyComment(CommentReply commentReply){
            commentReply.setContent(content);
            Long cmtId = Long.valueOf(commentId);
            commentReply.setCommentId(cmtId);
            commentReply.setReplyType((byte) 0);
            SentenceComment sc = commentDao.getCommentInfoById(cmtId);
            commentReply.setReplyCommentUserId(sc.getUserId());
            commentReply.setReplyObjectUserId(sc.getUserId());
            commentReply.setReplyObjectId(cmtId);
            UserInfo userInfo = (UserInfo) session.get("userInfo");
            commentReply.setReplyWriterId(userInfo.getId());
            commentReply.setReplyTime(new
Timestamp(System.currentTimeMillis()));
            commentDao.publishReply(commentReply);
        }

        /**
        * @author hasee
        * @Description 回复回复
        * @Date 16:00 2018-07-10
        * @Param [commentReply]
        * @return void
        **/
```

```
    public void replyReply(CommentReply commentReply){
        commentReply.setContent(content);
        long rpOId = Long.valueOf(rpObjId);
        CommentReply cr = commentDao.getCommentReplyInfoById(rpOId);
        commentReply.setCommentId(cr.getCommentId());
        commentReply.setReplyType((byte) 1);
        commentReply.setReplyCommentUserId(cr.getReplyCommentUserId());
        commentReply.setReplyObjectUserId(cr.getReplyWriterId());
        commentReply.setReplyObjectId(cr.getId());
        UserInfo userInfo = (UserInfo) session.get("userInfo");
        commentReply.setReplyWriterId(userInfo.getId());
        commentReply.setReplyTime(new
Timestamp(System.currentTimeMillis()));
        commentDao.publishReply(commentReply);
    }


    /**
    * @author hasee
    * @Description  数据合法性检验
    * @Date 15:47 2018-07-10
    * @Param []
    * @return java.lang.Boolean
    **/
    public Boolean dataValidationCheck(){
        UserInfo userInfo = (UserInfo) session.get("userInfo");
        if(userInfo == null){
            return false;
        }

        if(type == null){
            return false;
        }
        int sType = -1;
        try{
            sType    = Integer.valueOf(type);
        }catch (Exception e){
            return false;
        }
        if(sType != 0 && sType != 1){
            return false;
        }
        if(sType == 0){
            try{
```

```
                long cmdId = Long.valueOf(commentId);
                if(! commentDao.checkIfCommentExistById(cmdId)){
                    return false;
                }
            }catch (Exception e){
                return false;
            }
        }else{
            try{
                long rpOId = Long.valueOf(rpObjId);
                if(! commentDao.checkIfReplyExistById(rpOId)){
                    return false;
                }
            }catch (Exception e){
                return false;
            }
        }
        if(content == null || "".equals(content)){
            return false;
        }
        return true;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getCommentId() {
        return commentId;
    }

    public void setCommentId(String commentId) {
        this.commentId = commentId;
    }

    public String getRpObjId() {
        return rpObjId;
    }

    public void setRpObjId(String rpObjId) {
```

```
        this.rpObjId = rpObjId;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public CommentDao getCommentDao() {
        return commentDao;
    }

    public void setCommentDao(CommentDao commentDao) {
        this.commentDao = commentDao;
    }

    public String getSentenceId() {
        return sentenceId;
    }

    public void setSentenceId(String sentenceId) {
        this.sentenceId = sentenceId;
    }

    @Override
    public void setSession(Map<String, Object> session) {
        this.session = session;
    }
}
```