本系统是一个基于JSP+SSH+MySQL技术的BBS论坛系统。系统分为前台功能模块和后台功能模块。其中系统前台功能主要实现了用户注册、用户登录、个人中心、搜索、帖子列表、发布帖子、回复帖子、发布需求信息（设置积分奖励）等功能；系统后台功能实现用户管理、帖子管理（帖子加精、帖子置顶和作为管理员对文章进行修改）

**功能需求分析**

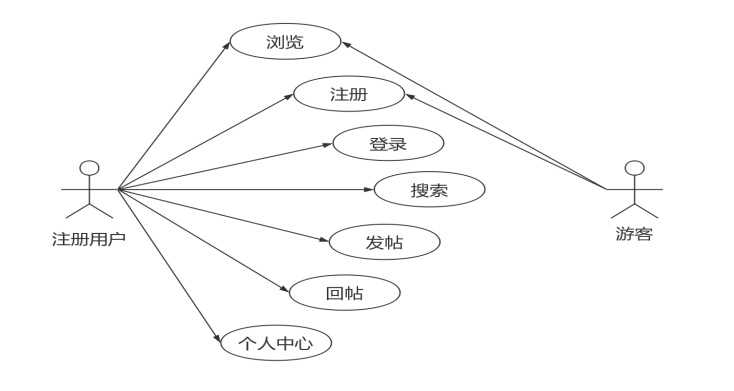
对于BBS论坛系统来说，最主要的功能就是用户浏览帖子、发布帖子以及回复帖子。同时考虑用户所发布帖子的内容需要健康且合法，这就需要管理员加强对帖子的审核。如果用户想要在论坛发布帖子，前提是需要注册且处于登录状态。此外，为了便于从内容上对帖子加以区分，方便管理员对帖子的管理，还需要将论坛分为多个版块。

综上，论坛既要实现对于帖子的管理，又要实现对于用户的管理，还要兼顾到各个功能模块之间的协调合作，共同完成BBS论坛系统的业务需求。下面采用UML用例模型对系统功能需求进行分析。

经过对BBS论坛系统的分析，可以确定本系统用例模型有两种，用户（用户分为游客和注册用户）和系统管理员。下面分别对这三个角色的功能进行描述：

1）用户

用户分为游客和注册用户，其中游客只能浏览论坛公共部分的内容，还可以注册成为论坛用户；注册用户则可以登录论坛、搜索帖子、发帖、回帖和查看个人中心。用户主要功能如下：



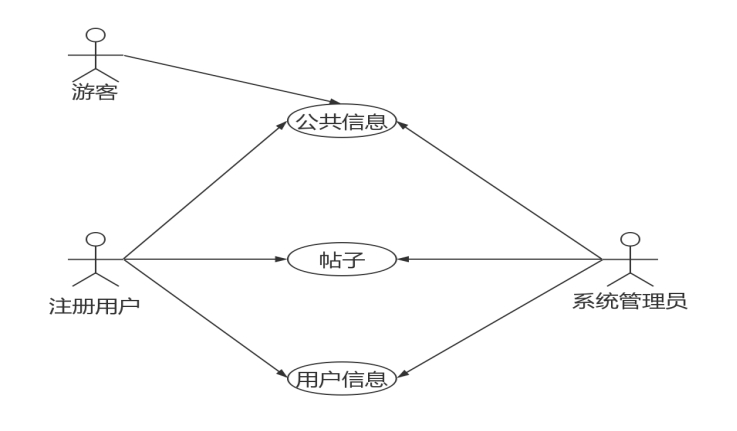
2）系统管理员

系统管理员主要负责系统的后台管理工作，主要功能如下管理员用例图）：

a. 用户管理（查看用户，删除用户）

b. 帖子管理（帖子删除，加精，置顶）；

确定了用户用例模型和管理员用例模型后，就可以构建BBS论坛系统的用例模型，整个系统的用例图如下图所示：



### 非功能需求分析

所谓非功能需求，是指软件产品为满足用户业务需求儿必须具有且除功能需求以外的特性。由于软件产品本身不是孤立存在的，还会受到外在环境中诸多因素的影响，因而非功能需求涉及的范围比较广。软件产品的非功能需求包括系统的实用性、稳定性和安全性等。非功能需求要求设计软件时，一方面要可用，另一方面要易用。

BBS论坛系统的非功能需求包括以下几个方面：

（1）系统的实用性

系统设计开发的功能要尽量简单、实用，用户在使用系统过程中能够感受到系统的快捷、方便，不存在很多累赘和多余的操作或者功能。

（2）系统的稳定性

系统稳定性差则预示着系统在用户使用时，可能会出现数据操作错误，页面响应时间过长或者干脆无法响应等，因此系统的稳定性是用户对系统评价的重要指标之一。

（3）系统的开放性

系统具有良好的可兼容性，可以在Windows的大部分版本的操作系统中正常的运行，另外，也支持不同浏览器版本，能够在微软的IE浏览器、谷歌浏览器等常用的浏览器上正常运行。系统的易升级和管理也是系统开放性的表现。

（4）系统的安全性

用户在身份认证、授权控制和私密性等方面要求较高的安全性。

**数据库**

本系统数据库中包括：1）管理员和用户表（t\_user） 2）版块表（t\_category）3）子版块表（t\_type）4）帖子表（t\_topic） 5）评论回复表（t\_comment）。各表数据结构如下：

（1）管理员和用户表(t\_user)，存储管理员和用户信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 是否空 | 说明 |
| id | int(11) | Y | N | 用户ID |
| username | varchar(20) | N | Y | 用户名 |
| password | varchar(20) | N | Y | 密码 |
| nickname | varchar(32) | N | Y | 昵称 |
| sex | varchar(2) | N | Y | 性别 |
| picture | varchar(255) | N | Y | 图片 |
| email | varchar(20) | N | Y | 邮箱 |
| come\_from | varchar(200) | N | Y | 地址 |
| introduction | longtext | N | Y | 简介 |
| profession | varchar(40) | N | Y | 职业 |
| grade\_integal | int(11) | N | N | 等级 |
| topic\_count | int(11) | N | Y | 帖子数量 |
| comment\_count | int(11) | N | Y | 评论回复数量 |
| role\_id | int(11) | N | N | 角色ID |
| users\_grade\_id | int(11) | N | N | 用户头衔ID |
| register\_time | datetime | N | Y | 注册时间 |
| status | int(11) | N | N | 状态 |

（2）版块表（t\_category），存储版块信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 是否空 | 说明 |
| id | int(11) | Y | N | 版块ID |
| name | varchar(30) | N | Y | 版块名称 |
| count\_topics | int(11) | N | N | 帖子数量 |
| count\_comments | int(11) | N | N | 评论回复数量 |

（3）子版块表（t\_type），存储子版块信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 是否空 | 说明 |
| id | int(11) | Y | N | 子版块ID |
| name | varchar(30) | N | Y | 子版块名称 |
| count\_topics | int(11) | N | N | 帖子数量 |
| count\_comments | int(11) | N | N | 评论回复数量 |
| is\_admin\_type | int(11) | N | N | 是否管理员类型 |
| types\_category\_id | int(11) | N | N | 版块ID |

（4）帖子表（t\_topic），存储帖子信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 是否空 | 说明 |
| id | int(11) | Y | N | 帖子ID |
| title | int(8) | N | Y | 帖子标题 |
| content | longtext | N | Y | 评论回复 |
| comment\_count | int(11) | N | N | 评论回复数量 |
| status | int(11) | N | N | 状态 |
| topic\_time | datetime | N | Y | 时间 |
| topics\_user\_id | int(11) | N | N | 用户ID |
| topics\_type\_id | int(11) | N | N | 子版块ID |
| integral | int(11) | N | Y | 是否完整 |

（5）评论回复表（t\_comment），存储评论回复信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主键 | 是否空 | 说明 |
| id | int(11) | Y | N | 评论回复ID |
| content | longtext | N | Y | 内容 |
| floor | int(11) | N | Y | 回复楼层 |
| comment\_time | datetime | N | Y | 时间 |
| comments\_user\_id | int(11) | N | N | 用户ID |
| comments\_topic\_id | int(11) | N | N | 帖子ID |
| integral | int(11) | N | N | 是否完整 |
| status | int(11) | N | N | 状态 |

**系统架构**

本系统采用SSH框架开发，系统架构主要分为四层：**表示层、业务逻辑层、数据持久层和实体层**。

1、在**表示层**中，首先通过JSP页面实现交互界面，负责传送请求(Request)和接收响应(Response)，然后Struts根据配置文件(struts-config.xml)将ActionServlet接收到的Request委派给相应的Action处理。

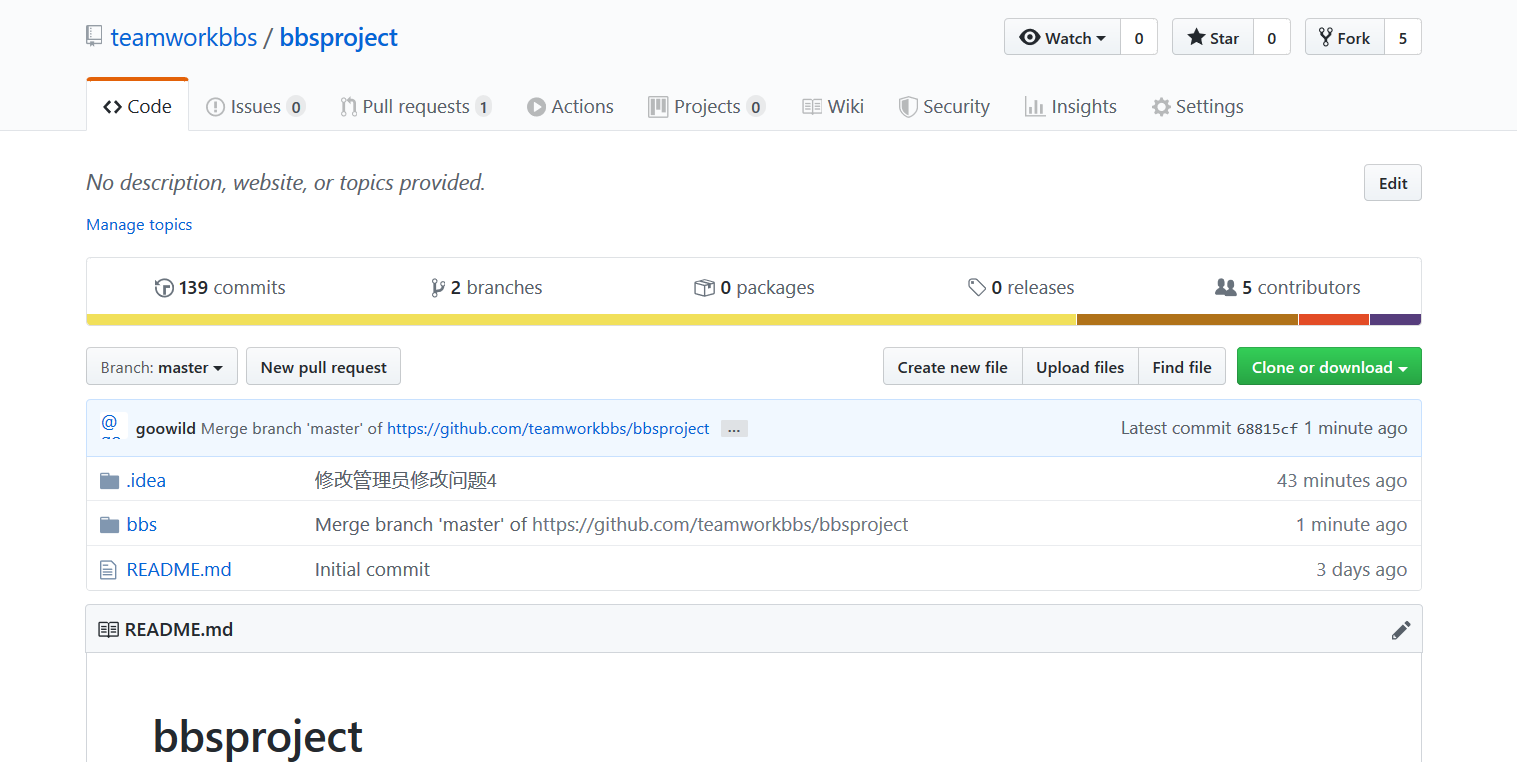
2、在**业务逻辑层**中，管理服务组件的Spring IoC容器负责向Action提供业务模型(Model)组件和该组件的协作对象数据处理(DAO)组件完成业务逻辑。

3、在**持久层**中，则依赖于Hibernate的对象化映射和数据库交互，处理DAO组件请求的数据，并返回处理结果。

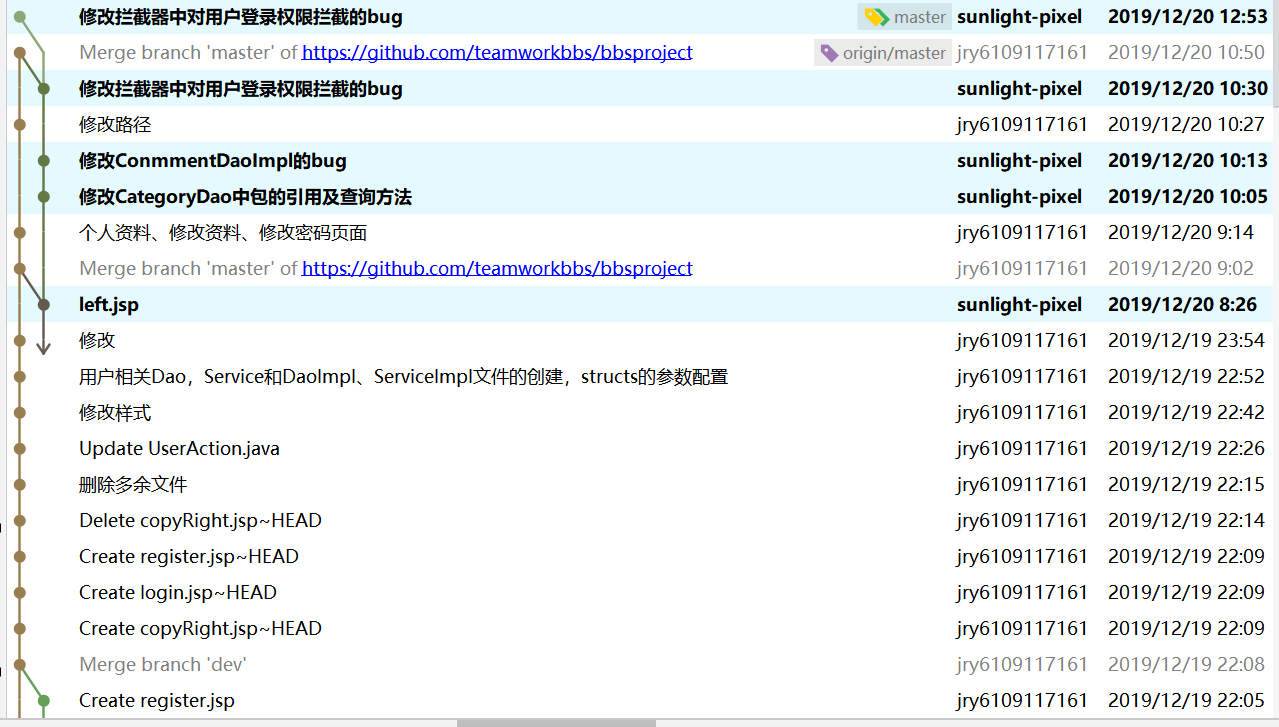
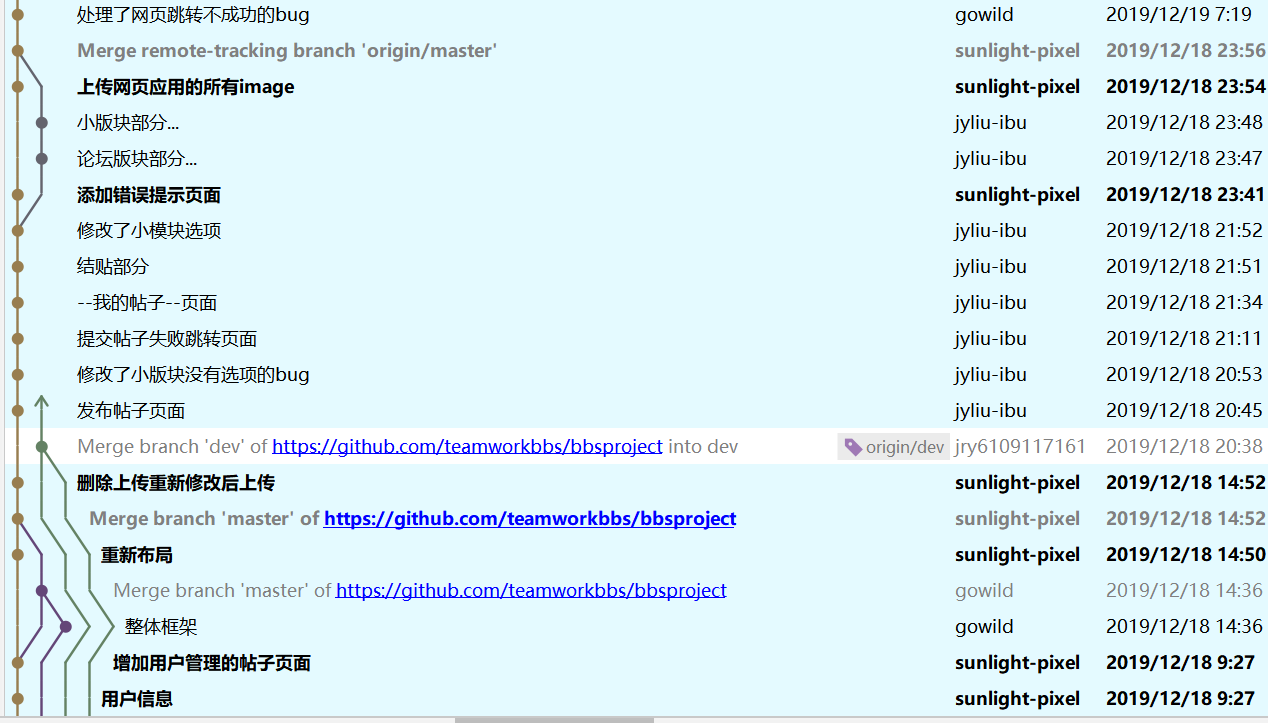
采用上述开发模型，不仅实现了视图、控制器与模型的彻底分离，而且还实现了业务逻辑层与持久层的分离。这样无论前端如何变化，模型层只需很少的改动，并且数据库的变化也不会对前端有所影响，大大提高了系统的可复用性。

**合作方式**

1. 在github上建立组织，上传文件到仓库。



2、通过git的基本命令对远程仓库内容进行更改和提交。提交截图（部分截图）如下：



流程如下图：

创建项目bbsproject->

导入相关jar包->

在/WebContent/WEB-INF/下新建一个web.xml，配置过滤器和监听器web.xml->在/Java Resources/src/目录下创建四个包com.bbs.bean、com.bbs.dao、com.bbs.service、com.bbs.action->

在.bean下创建Users实体类和与之对应的hibernate的配置文件Users.hbm.xlm文件->

在.dao下创建接口文件UserDao.java及实现其对数据库增删改查方法的UserDaoImpl.java文件->

在.service下创建UserService.java接口和实现接口方法的类UserServiceImpl.java文件->

在.action下创建UserAction类对表单传过来数据经过相应功能分析后再据此设置接下来要跳转的页面->

编写struts的配置文件struts.xml->

编写Spring和hibernatep配置文件applicationContext-db.xml（SSH框架基础搭建成功）

**组员分工**

注册登录修改信息-江如烟

发布回复帖子设置积分-杨方嘉、刘瑾瑶

帖子置顶加精-王春阳

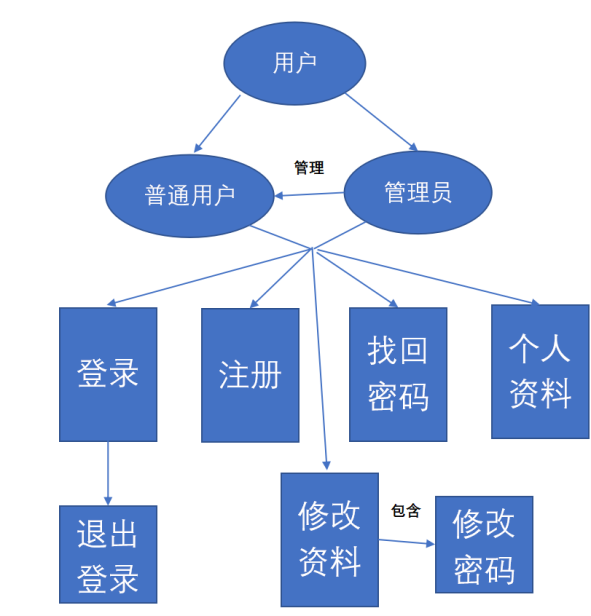
用户对自己的帖子进行操作-刘燕玲

初始时：

创建用户表，用户有以下属性：id编号（主键）、用户名称、密码、昵称、性别、头像、邮箱、地点、个人介绍、职业、权限（管理员与普通用户）。

在后续与小组人员交互中，新增添了未读信息、个人积分、帖子数、评论数、个人等级、注册时间、及账号是否被禁用这些属性。

负责完成的功能如图：



登录：验证用户名和密码格式是否正确，检查数据库中是否有对应用户，有则登录成功，否则失败。

注册：验证输入信息格式是否合法，把新用户信息插入数据库中。

找回密码：验证输入用户名和邮箱格式是否正确，检查数据库中是否有对应用户，有则返回用户密码，否则失败。

个人资料：根据用户id获取用户相关信息显示在页面上。

修改资料：验证输入信息格式是否合法，根据输入的用户信息修改数据库中用户的资料。

修改密码：验证输入密码格式是否合法、旧密码是否输入正确，修改后密码与旧密码应不一致，把修改后的密码插入数据库中。

用户可以session来获取，具有时效性，会话过期后获取的用户信息失效。