**It 项目管理报告书**

项目名称： 校园论坛

小 组： 组 长： 组 员：

**2016** 年 **12** 月 **18** 日

目录

[It 项目管理报告书 1](#_bookmark0)

[1 成员任命书 4](#_bookmark1)

[1.2 分工表 5](#_bookmark2)

* 1. [系统概述 5](#_bookmark2)
  2. [系统目标 6](#_bookmark3)
  3. [文档概述 7](#_bookmark4)
  4. [需求概述 7](#_bookmark4)
  5. [功能需求 9](#_bookmark5)
  6. [系统用例图 13](#_bookmark6)
  7. [系统时序图 14](#_bookmark7)
  8. [系统组件图 15](#_bookmark8)
  9. [系统 ER 图 16](#_bookmark9)
  10. [系统配置图 17](#_bookmark10)

1. [项目需求变更 17](#_bookmark10)
2. [项目周报 18](#_bookmark11)
3. [操作指引 20](#_bookmark12)
   1. [项目简介 21](#_bookmark13)
   2. [操作介绍 22](#_bookmark14)
4. [业务说明 25](#_bookmark15)
   1. [业务简介 25](#_bookmark15)
   2. [业务场景 26](#_bookmark16)
   3. [业务实现代码 28](#_bookmark17)
5. [数据库 96](#_bookmark18)

[数据流图 96](#_bookmark18)

* 1. [顶层数据流图 97](#_bookmark19)
  2. [0 层数据流图 97](#_bookmark19)
  3. [具体数据流图 98](#_bookmark20)
  4. [数据字典 99](#_bookmark21)

[7.5 ER 图 103](#_bookmark22)

[7.6 数据库设计 104](#_bookmark23)

1. [需求新增 109](#_bookmark24)
   1. [初始需求 109](#_bookmark24)
   2. [第一次新增需求 110](#_bookmark25)
   3. [第二次新增需求 110](#_bookmark25)

[9.功能确认 110](#_bookmark25)

* 1. [测试目的 110](#_bookmark25)
  2. [测试人员 111](#_bookmark26)
  3. [测试环境 111](#_bookmark26)
  4. [测试内容 111](#_bookmark26)
  5. [功能截图 112](#_bookmark27)
  6. [测试结论 117](#_bookmark28)

1. [逻辑确认 117](#_bookmark28)

[10.1 表示层 117](#_bookmark28)

[10.2 业务逻辑层 119](#_bookmark29)

1. [验收报告 121](#_bookmark30)
   1. [项目基本情况 122](#_bookmark31)
   2. [项目概述 122](#_bookmark31)
   3. [验收测试环境 122](#_bookmark31)
   4. [验收及测试结果 124](#_bookmark32)
   5. [验收总结 125](#_bookmark33)
   6. [参考资料 125](#_bookmark33)

**1.** 成员任命书

为完成 it 项目管理大作业的需要，由本人第\_\_\_\_小组组长决定，对以下成员进行任命

任命 为项目组长，负责前端开发以及组内的工作安排任命 为文档编写员，负责需求分析，会议记录，文档编 写

任命 为总体设计员，负责总体设计，uml 类图，ER 图等的设计

任命 为详细设计员，负责详细设计，UI 设计

任命 为后端开发员，负责数据库构建和后端开发

任命 为软件测试员，负责进行软件测试

任命人：

时间：2016 年 10 月 6 日

**1.2** 分工表

|  |  |
| --- | --- |
| 组长 |  |
| 文档 |  |
| 设计 |  |
| 前端 |  |
| 后端 |  |
| 测试 |  |

* 1. 系统概述
     1. 网站名称：

校园论坛。

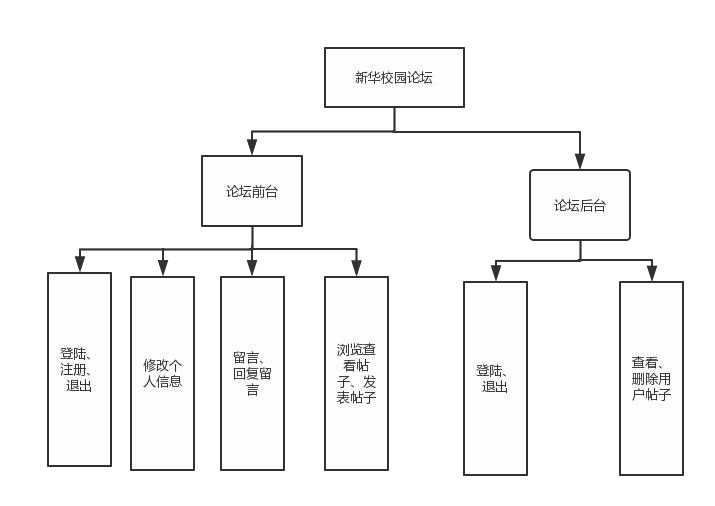
* + 1. 网站功能实现：

为用户提供一个注册、发帖、回复、浏览等交流操作功能。

* + 1. 用户：

我校学生

* + 1. 子系统关系图：



* + 1. 开发者：
  1. 系统目标

为了方便校内同学的交流，我们决定要做这么一个校园论坛，而对于论坛这样的数据流量特别大的网络管理系统， 必须要满足使用方便、操作灵活等设计需求。所以本系统在设计时应满足以下几个目标：

**1.**临时用户进入**,**可浏览帖子但不可发帖

1. 一个版面能显示所有的帖子
2. 具有登录模块，有用户的个人信息
3. 用户随时都可以查看自己发表的帖子
4. 有用户的消息的时间及时提醒，主要用于提示哪个用户 回复了自己的主题。 6.管理员权限可删除任意帖子，具有最大权限的管理功能
5. 对用户输入的数据，系统进行严格的数据检验，尽可能 排除人为的错误
6. 系统最大限度地实现了易维护性和易操作性
7. 系统运行稳定安全可靠
   1. 文档概述

需求分析报告采用面向对象的方法，在文档中主要采用了用例、流程图等表示方法来描述需求。

* 1. 需求概述
     1. 用户需求

对于一个用户，使用论坛进行交流时，首先要注册一个 账户，然后登录后才能进行对帖子的回复，如果不登录，就 只能查看帖子而不能进行回复和发表帖子。用户使用论坛系 统的需求是发表某一个主题相关的帖子，用户在发表帖子后， 如果有人进行回复，就要在首页提醒用户有新消息。用户可

以删除自己发表的帖子和评论。对于论坛管理人员来说，需

要完成对用户发表的帖子的管理，包括：设置精华帖、置顶 帖子、删除帖子等操作。

开发校园论坛系统的目的是提供一个供我校学生交流的平台，为我校学生提供交流经验、探讨问题的社区。因此， 校园论坛系统最基本的功能首先是发表主题，其次是其他人员根据主题发表自己的看法。此外，为了记录主题的发表者和主题的回复者信息，系统还需要提供用户注册和登录的功能。只有注册的用户登录后才能够发表和回复主题，浏览者

（游客）只能浏览主题信息。根据用户的需求及以上的分析， 校园论坛需要具备前台功能和后台功能。

**1)**系统前台功能：显示用户发表的帖子，查看帖子的内 容、发表对帖子的回复、发表对回复的回复、显示用户的头像、用户信息的显示、用户新信息的提醒。

**2)**系统后台功能：进入后台、帖子管理、用户管理、添 加删除用户、系统设置、退出系统、返回首页。

* + 1. 系统开发环境需求
       1. 开发环境

我们一致认为在开发此论坛的时候需要配置以下软件 环境：

服务器端：

(1)操作系统：Windows 7 及以上；

(2)Web服务器：Tomcat 7.0 ；

(3）集成开发环境：Eclipse ；

(4）数据库：MySQL 5.5； 客户端

无特别要求

* + - 1. 使用技术

前端：HTML、CSS、JS、BootStrap

后端：Java

数据库：MySQL

* + - 1. 用户的特点

本网站的最终用户的特点：所有上网用户在无需培训的情况下，按照网站页面提示即可使用网站的相关服务和功能 后台管理和维护人员的教育水平和技术专长：本软件的

后台管理和维护人员均为我小组开发成员。

* 1. 功能需求

经过系统性的研究，我们认为整个论坛大概分为 3 个功能模块，分别为论坛模块，管理员模块和用户模块

* + 1. 前台功能需求

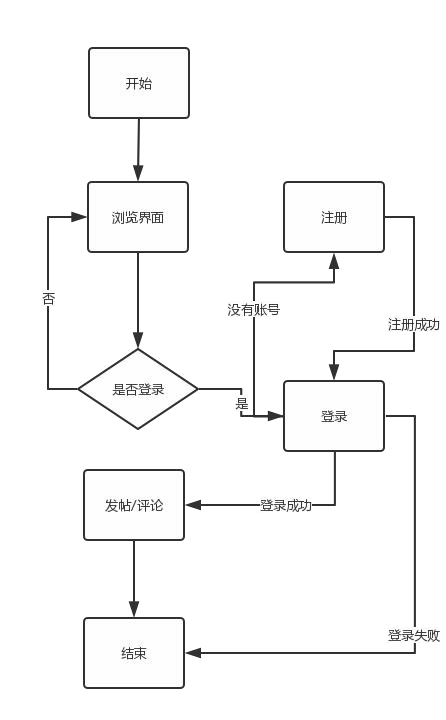
在论坛前台中，我们作了如下设计：分未登录前和登录后，登录前，用户进入校园论坛的主页面后，可以查看帖子内容、用户注册、用户登录，登录后，用户可以修改个人信息、查看个新消息、修改头像、查看帖子内容、回复帖子。

* + 1. 后台功能需求

管理用户进入后台后，可以进行帖子管理，包括查看帖子、删除帖子、返回论坛首页和退出系统。

* + 1. 系统前台流程图

在系统前台流程中，我们做出了如下设置。首先，我们开始点开界面，是我们的论坛主页，不登录可以以临时用户身份浏览，登陆则可以发帖和评论，没有账号的可以注册。



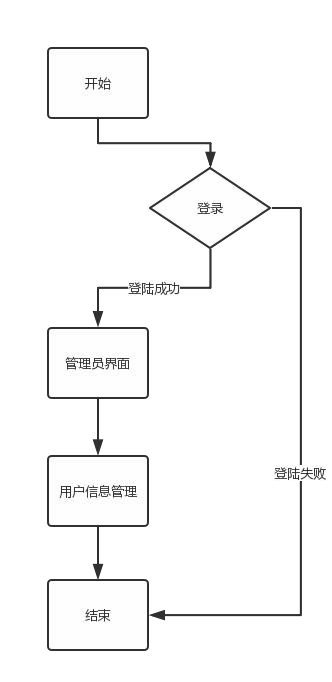
系统前台流程图

* + 1. 系统后台流程图

在系统的后台流程中，我们做出了如下设置。首先，我

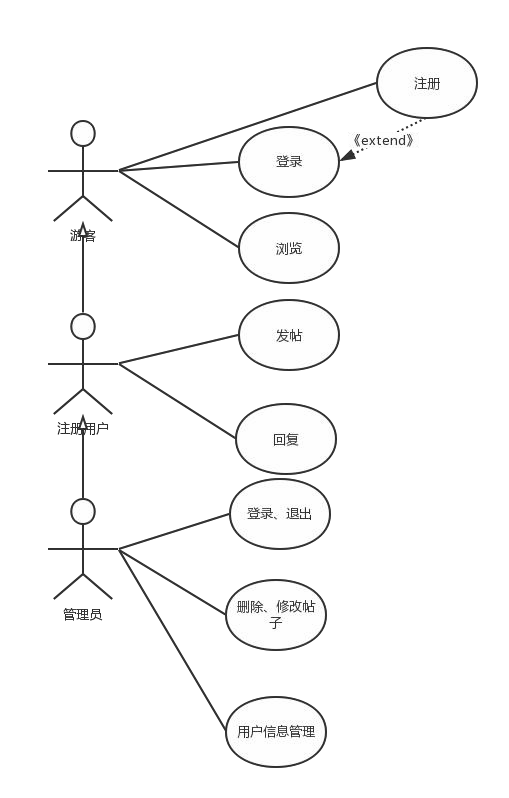
们进入登录界面，使用管理员账号和密码进行登录，在管理

员界面，我们可以进行

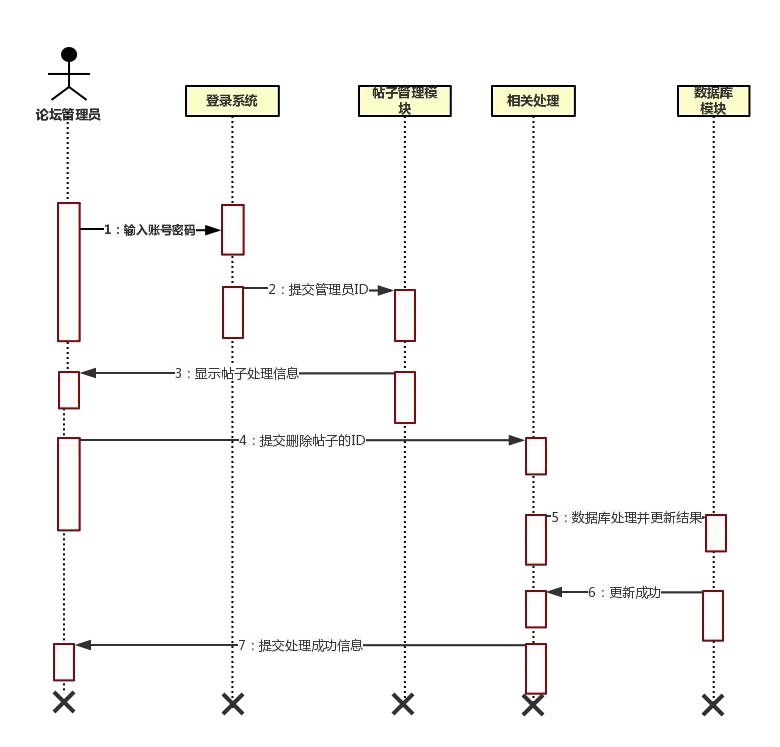
用户信息管理，可以查看、删除用户帖子

系统后台流程图

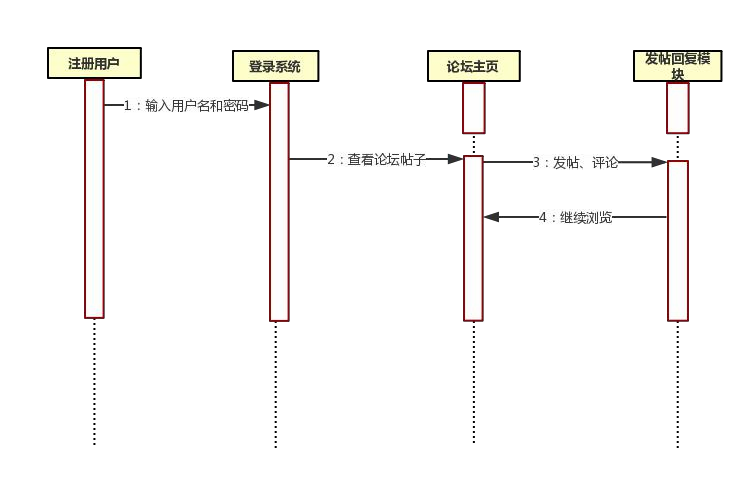
* 1. 系统用例图



* 1. 系统时序图

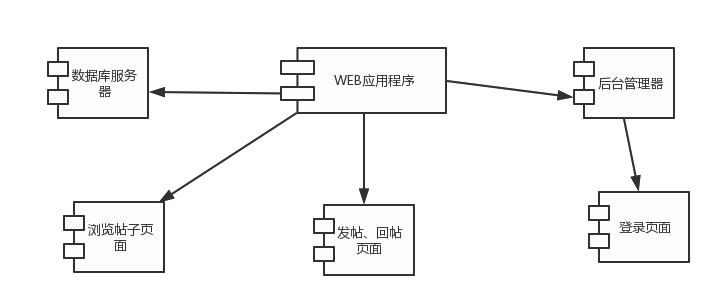


论坛管理员处理帖子的时序图

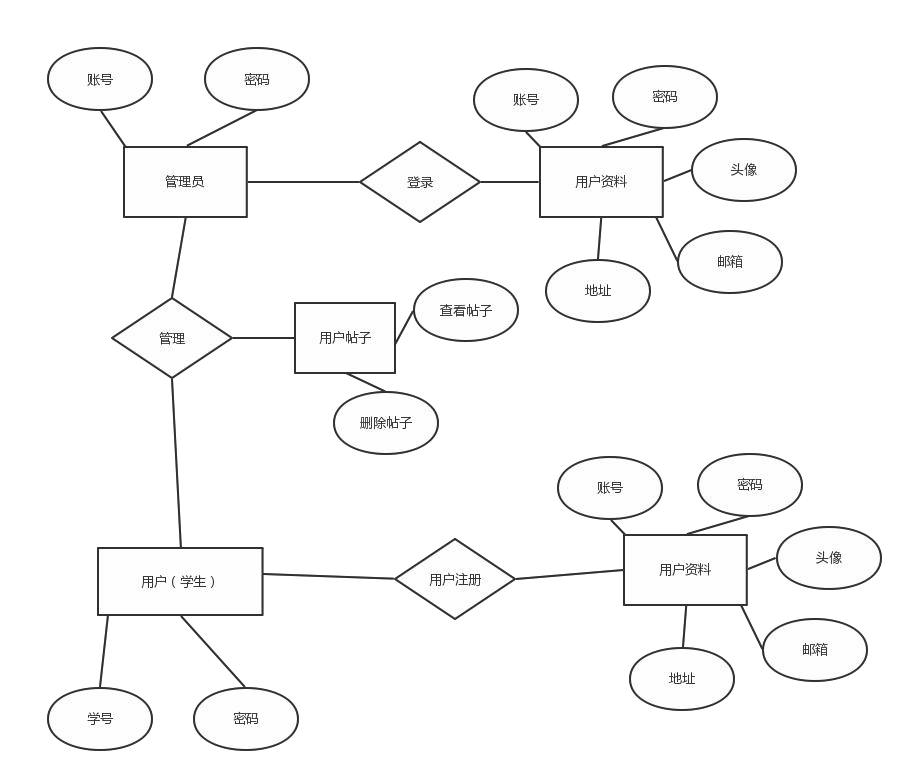


用户发帖评论时序图

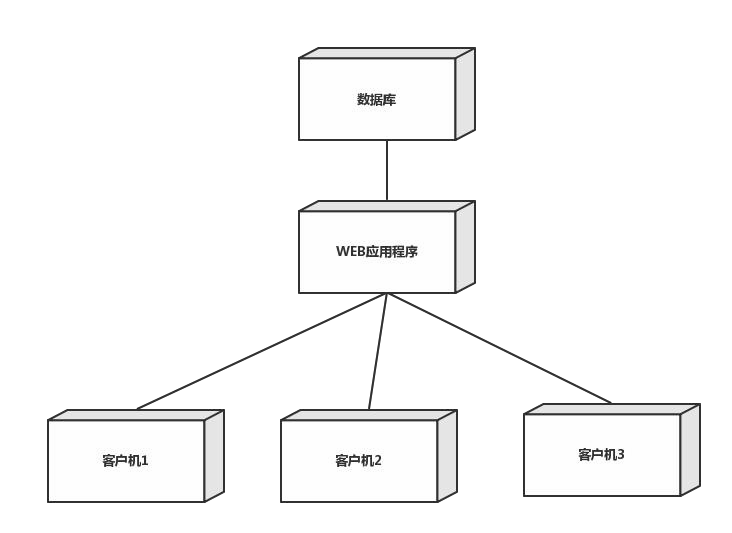
* 1. 系统组件图



* 1. 系统 **ER** 图



* 1. 系统配置图



1. 项目需求变更

|  |  |  |  |
| --- | --- | --- | --- |
| 变更名称 | 新增验证码校验功能 | | |
| 变更时间 | 2016-10-28 | 变更申请人 |  |
| 变更类型 | 验证码校验 | | |
| 变更理由 | 增强系统的安全性，防止有恶意软件在注册界面持续不断注册信息，导致系统瘫痪。 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 变更名称 | 新增 md5 加密功能 | | |
| 变更时间 | 2016-11-5 | 变更申请人 |  |
| 变更类型 | md5 加密 | | |
| 变更理由 | 增强用户数据的安全性，在用户注册、用户登录界面，数据 库存储中起作用，防止客户端在向服务器发送请求的过程 中，被他人截获明文密码；另外，对数据库管理人员也有防范作用，使得用户账户更安全。 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 变更名称 | 添加页脚 | | |
| 变更时间 | 2016-11-5 | 变更申请人 |  |
| 变更类型 | 前端开发 | | |
| 变更理由 | 提供网页制作者信息以及反馈渠道 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 变更名称 | 新增图片轮播功能 | | |
| 变更时间 | 2016-11-5 | 变更申请人 |  |
| 变更类型 | md5 加密 | | |
| 变更理由 | 增强观赏性，且可用于宣传或者广告 | | |

1. 项目周报

|  |  |
| --- | --- |
| 项目 | 校园论坛 |

|  |  |
| --- | --- |
| 报告时间 | 第六周 |
| 报告人 |  |
| 报告  内容 | 1. 进行系统设计 2. 进行详细设计 3. 进行项目的计划设计 |

|  |  |
| --- | --- |
| 项目 | 校园论坛 |
| 报告时间 | 第八周 |
| 报告人 |  |
| 报告内容 | 1. UI 界面的设计 2. 数据库的设计 |

|  |  |
| --- | --- |
| 项目 | 校园论坛 |
| 报告时间 | 第九周 |

|  |  |
| --- | --- |
| 报告人 |  |
| 报告内容 | 1. 制作未包含样式的程序 2. 重新进行数据库的设计 3. 尝试使用MD5算法进行密码加密 |

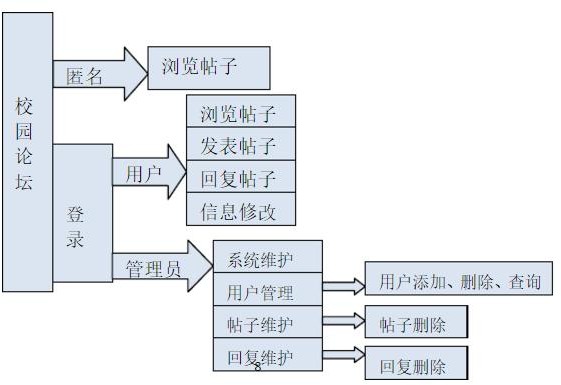
|  |  |
| --- | --- |
| 项目 | 校园论坛 |
| 报告时间 | 第十周 |
| 报告人 |  |
| 报告内容 | 1.对样式的添加  2.添加注册时的验证码  3.采用 js 进行表单验证  4.采用 MD5 对密码进行加密 |

|  |  |
| --- | --- |
| 项目 | 校园论坛 |
| 报告时间 | 第十一周 |
| 报告人 |  |
| 报告内容 | 1.使用 Bootstrap 对页面进行美化  2.添加轮播的功能 |

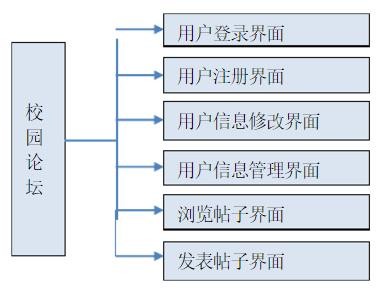
1. 操作指引
   1. 项目简介

校园论坛所具有的功能包括：用户注册、用户登录、用户信息修改、浏览帖子、发表帖子、收藏帖子、搜索帖子、回复帖子、用户信息管理（查询、增加、删除、修改）。

从整体上可以分为数据层、数据访问层和业务逻辑层。数据层是系统最底层，它用于存储系统的所有数据。数据访问层建立在数据库之上，应用程序通过该层访问数据库。数据访问层一般封装数据库的选择、添加、更新和删除操作， 同时还为业务逻辑层服务，所以数据访问层的设计的好坏关系到整个系统的成败。业务逻辑层包括用户登录、用户注册、 发表帖子等业务逻辑，它一般由Web页面实现。

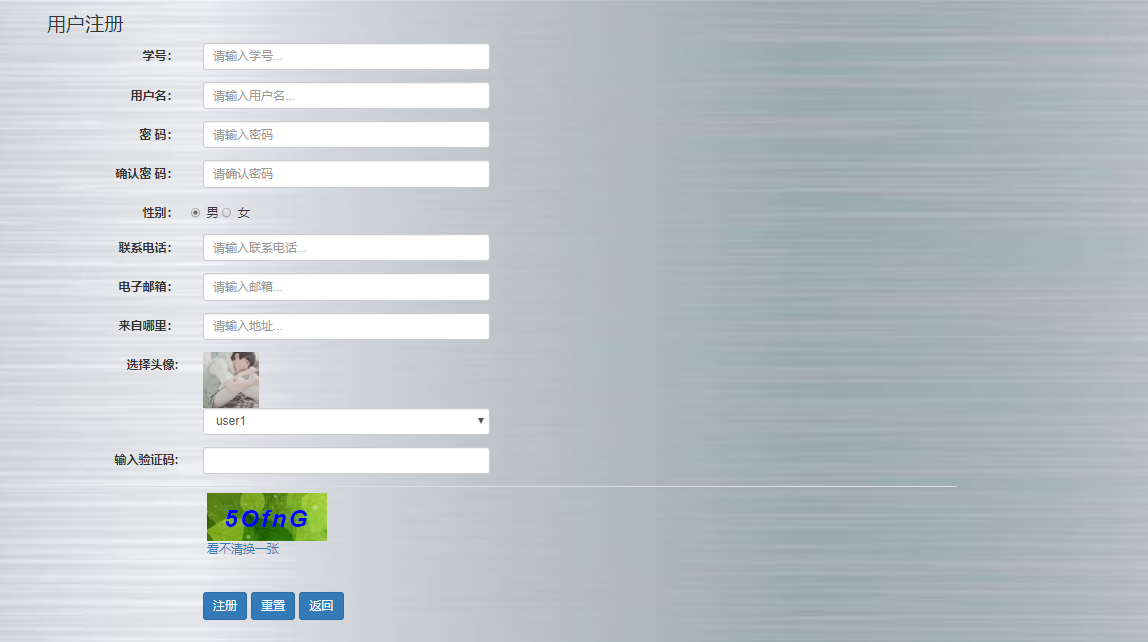
（1）系统操作结构

（2）页面操作结构



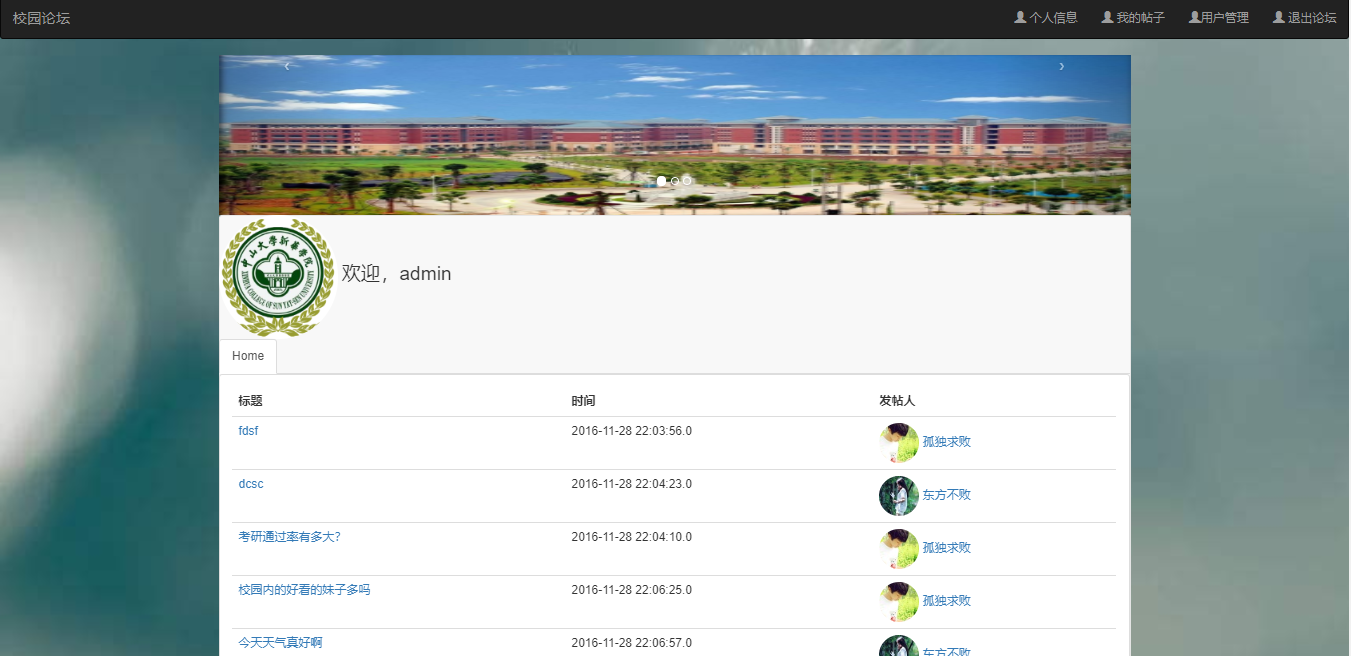
* 1. 操作介绍

在登录注册界面可以通过登录和注册按钮进行登录和 注册操作



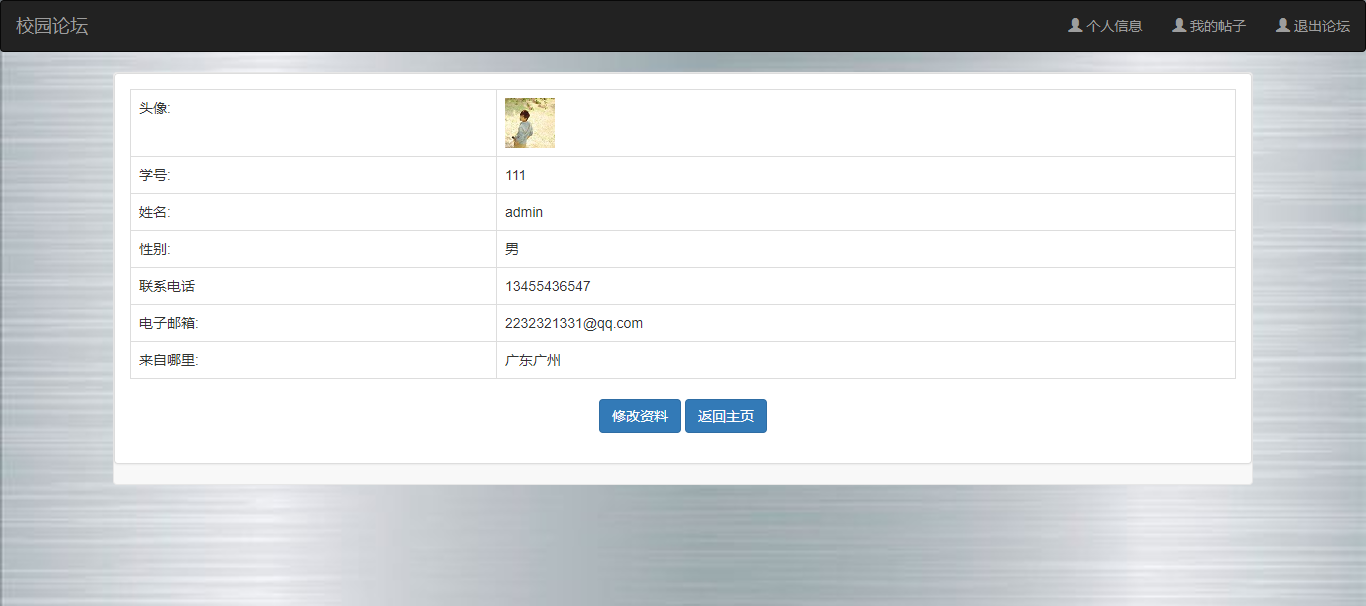
登录完就会进入主界面，在主界面上方有“个人信息”，

“我的帖子”、“用户管理”等按钮可以进行相应的操作。界面中间是其他用户发的帖子，可以点击进行浏览和恢复等操作。界面的最下方是发帖模块，只用登录用户才可以进行发 帖操作，游客只有浏览帖子的权限。



点击个人信息按钮进入个人信息界面可以修改个人的

详细信息。



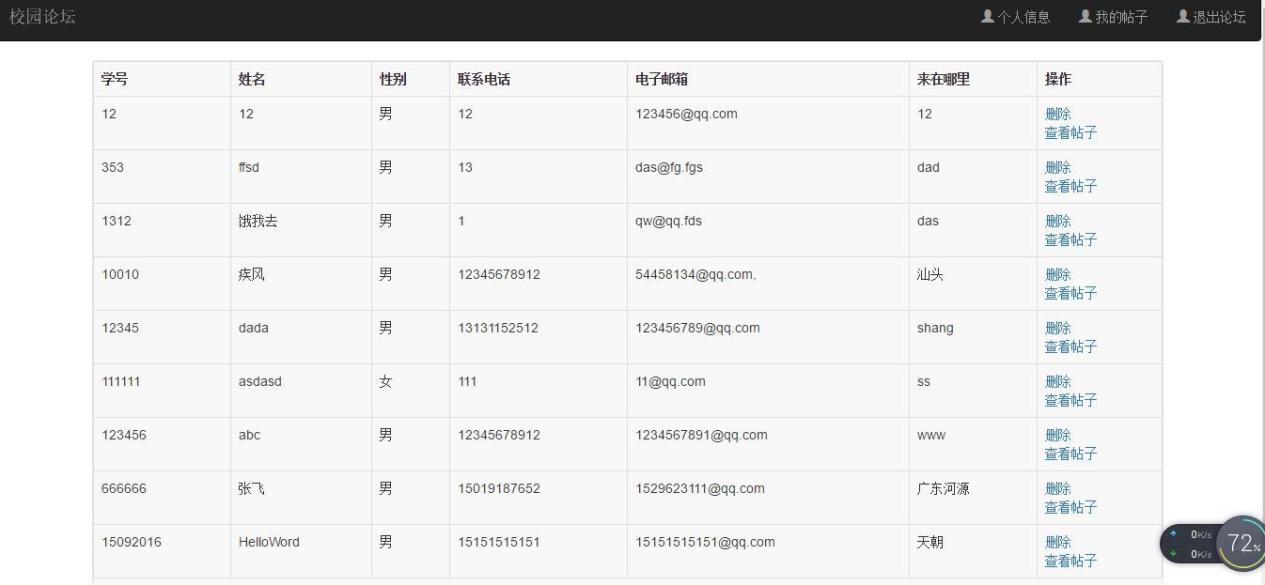
点击我的帖子进入我的帖子界面可以对自己发的帖子 进行删除和查看操作。



在首页点击其他用户的帖子可以进入帖子的完整内容 进行浏览，还可以在最下方的回复模块进行回复。



如果你是以管理员的身份登录，你还可以进入用户管理

模块，进行删除帖子的操作

1. 业务说明
   1. 业务简介

本软件系统旨在通过网络论坛，让在校大学生快速地进

行交流更为便捷。使得大学生的交流环境和校方教育及管理

人员获得广大学生声音更加方便也更为人性化。校园论坛是面向各大高校师生的一个信息交流平台，建立一个基于师生沟通为目的，功能简洁，方便实用和管理的论坛系统显得越来越必要。为达成这一目标，并将大学学习期间所学的数据库设计与实现、网页设计、面向对象程序设计、Web 应用开发等知识综合运用，融会贯通，特开发此项目

* 1. 业务场景

（**1**）

|  |  |
| --- | --- |
| 触发事件 | 用户注册 |
| 执行者 | 用户 |
| 工作内容 | 1.进行用户的信息注册 |

（**2**）

|  |  |
| --- | --- |
| 触发事件 | 用户登录 |
| 执行者 | 用户 |
| 工作内容 | 1.用户使用已注册的账号和密码进行登录 |

（**3**）

|  |  |
| --- | --- |
| 触发事件 | 查看已发布的帖子 |
| 执行者 | 用户/游客 |

|  |  |
| --- | --- |
| 工作内容 | 1. 对已发布的帖子进行查看 |

（**4**）

|  |  |
| --- | --- |
| 触发事件 | 发帖 |
| 执行者 | 用户 |
| 工作内容 | 1.用户进行帖子发布 |

（**5**）

|  |  |
| --- | --- |
| 触发事件 | 回帖 |
| 执行者 | 用户 |
| 工作内容 | 1.用户对已发布的帖子内容进行回复 |

（**6**）

|  |  |
| --- | --- |
| 触发事件 | 论坛出现违规帖子 |
| 执行者 | 网站管理员 |
| 工作内容 | 1. 对违规帖子进行查看，评估   3.对存在违规现象的帖子进行删除，当情况严重时还需要对违规用户进行禁言或封号处理 |

* 1. 业务实现代码
     1. 注册实现

1. servlet

@SuppressWarnings("serial")

**public class** RegisterServlet **extends** HttpServlet{

**public void** service(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

// String user\_id = request.getParameter("user\_id");

String userid = request.getParameter("user\_id");

**int** user\_id=Integer.*parseInt*(userid); String user\_name =

request.getParameter("user\_name"); String user\_password =

request.getParameter("md5Pwd");

String user\_sex = request.getParameter("user\_sex");

String user\_face = request.getParameter("user\_face");

String user\_phone = request.getParameter("user\_phone");

String user\_email = request.getParameter("user\_email");

String user\_from = request.getParameter("user\_from");

UserEntity user = **new** UserEntity(); user.setUser\_id(user\_id);

user.setUser\_name(user\_name); user.setUser\_password(user\_password); user.setUser\_sex(user\_sex); user.setUser\_face(user\_face); user.setUser\_phone(user\_phone); user.setUser\_email(user\_email); user.setUser\_from(user\_from);

UserServiceInf service = **new**

UserServiceImpl();

service.saveUser(user);

//PrintWriter out=response.getWriter();

response.sendRedirect("login.jsp");

//out.print("<a href='login.jsp'>注册成功，请返回登录界面</a>");

}

}

1. ServiceInf

**public interface** UserServiceInf {

// 添加用户

**public int** saveUser(UserEntity user);

}

1. ServiceImpl

**public class** UserServiceImpl **implements**

UserServiceInf{

**public int** saveUser(UserEntity user) {

//调用dao保存用户

UserDaoInf userDao = **new** UserDaoImpl();

**return** userDao.addUser(user);

}

**public** UserEntity findUser(**int** uid){ UserDaoInf userDao=**new** UserDaoImpl(); UserEntity user=userDao.findUserById(uid); **return** (user==**null**)?**null**:user;

}

**public** List<UserEntity> findUserList(){

UserDaoInf userDao=**new** UserDaoImpl();

List<UserEntity> userList=userDao.findUserlist();

**return** userList;

}

}

1. DaoImpl

**public int** addUser(UserEntity user) { Connection conn = **null**; PreparedStatement ps = **null**;

String sql = "insert into user\_info(user\_id,user\_name,user\_password,user\_ sex,user\_face,user\_phone,user\_email,user\_from) values(?,?,?,?,?,?,?,?)";

System.*out*.println(user.getUser\_id());

**int** i = 0;

**try**{

// String newPassword=MD5.generateCode(user.getUser\_passw ord());

conn = DBUtil.*getConnection*();

ps = conn.prepareStatement(sql); ps.setInt(1, user.getUser\_id()); ps.setString(2, user.getUser\_name()); ps.setString(3,

user.getUser\_password());

ps.setString(4, user.getUser\_sex()); ps.setString(5, user.getUser\_face()); ps.setString(6, user.getUser\_phone()); ps.setString(7, user.getUser\_email()); ps.setString(8, user.getUser\_from());

i = ps.executeUpdate();

System.*out*.println("插入数据库成功");

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(**null**,ps, conn);

}

**return** i;

}

* + 1. 登录实现

1. servlet

@SuppressWarnings("serial")

**public class** LoginServlet **extends** HttpServlet {

**public void** service(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8"); String user\_id =

request.getParameter("user\_id"); String user\_password =

request.getParameter("md5Pwd");

UserServiceInf service = **new**

UserServiceImpl();

UserEntity user = service.login(user\_id, user\_password);

response.setContentType("text/html"); PrintWriter out = response.getWriter();

out.println("<link rel='stylesheet' type='text/css' href='css/easyui.css' />");

out.println("<link rel='stylesheet' type='text/css' href='css/icon.css' />");

out.println("<script type='text/javascript'

src='js/jquery-1.11.2.js'></script>"); out.println("<script

type='text/javascript' src='js/jquery.easyui.min.js'></script>");

**if**(user!=**null**){

request.getSession().setAttribute("user", user);

response.sendRedirect("forumList");

}**else**{

String html = "<div id='confirmdialog' class='easyui-dialog' title='消息确认' style='width:300px;height:160px;padding:10px'";

html = html + "data-options=\"iconCls: 'icon-no',buttons: [{text:'确定

',iconCls:'icon-ok',handler:function(){$('#conf

irmdialog').dialog('close');window.location.hre f='login.jsp';}}]\"";

html = html + ">用户名或密码有错！</div>";

out.print(html);

}

out.flush();

out.close();

}

}

1. service

**public class** UserServiceImpl **implements**

UserServiceInf{

**public** UserEntity login(String userId, String

passwd) {

//调用dao做用户的查询

UserDaoInf userDao = **new** UserDaoImpl(); UserEntity user = userDao.findUser(userId,

passwd);

**return** (user==**null**) ? **null** : user;

}

**public void** updateUser(UserEntity user){ UserDaoInf userdao=**new** UserDaoImpl(); userdao.updateUser(user);

}

**public** UserEntity findUser(**int** uid){ UserDaoInf userDao=**new** UserDaoImpl(); UserEntity user=userDao.findUserById(uid); **return** (user==**null**)?**null**:user;

}

**public** List<UserEntity> findUserList(){ UserDaoInf userDao=**new** UserDaoImpl();

List<UserEntity> userList=userDao.findUserlist();

**return** userList;

}

}

1. DaoImpl

**public class** UserDaoImpl **implements** UserDaoInf{

**public** UserEntity findUser(String userId, String passwd) {

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs = **null**;

String sql = "select \* from user\_info where user\_id=? and user\_password=?";

UserEntity user = **null**; **try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setString(1, userId); ps.setString(2, passwd);

rs = ps.executeQuery();

**if**(rs.next()){

user = **new** UserEntity();

user.setUser\_id(rs.getInt(1)); user.setUser\_name(rs.getString(2));

user.setUser\_password(rs.getString(3)); user.setUser\_sex(rs.getString(4)); user.setUser\_face(rs.getString(5)); user.setUser\_phone(rs.getString(6)); user.setUser\_email(rs.getString(7)); user.setUser\_from(rs.getString(8)); user.setIsAdmin(rs.getInt(9));

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** user;

}

}

**public** UserEntity findUserById(**int** id) { Connection conn = **null**; PreparedStatement ps = **null**;

ResultSet rs = **null**;

String sql = "select \* from user\_info where user\_id=?";

UserEntity user = **null**; **try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setInt(1,id);

rs = ps.executeQuery();

**if**(rs.next()){

user = **new** UserEntity(); user.setUser\_id(rs.getInt(1)); user.setUser\_name(rs.getString(2));

user.setUser\_password(rs.getString(3)); user.setUser\_sex(rs.getString(4)); user.setUser\_face(rs.getString(5)); user.setUser\_phone(rs.getString(6)); user.setUser\_email(rs.getString(7)); user.setUser\_from(rs.getString(8));

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** user;

}

**public** List<UserEntity> findUserlist(){ Connection conn=**null**; PreparedStatement ps=**null**;

ResultSet rs=**null**;

List<UserEntity> userlist=**new**

ArrayList<UserEntity>();

String sql="select \* from user\_info";

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); rs=ps.executeQuery();

**while**(rs.next()){

UserEntity user=**new** UserEntity(); user.setUser\_id(rs.getInt(1)); user.setUser\_name(rs.getString(2));

user.setUser\_password(rs.getString(3)); user.setUser\_sex(rs.getString(4)); user.setUser\_face(rs.getString(5)); user.setUser\_phone(rs.getString(6)); user.setUser\_email(rs.getString(7)); user.setUser\_from(rs.getString(8)); user.setIsAdmin(rs.getInt(9));

userlist.add(user);

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** userlist;

}

}

* + 1. 查看帖子

1. servlet

@SuppressWarnings("serial")

**public class** ForumListServlet **extends**

HttpServlet{

**public void** service(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException { BbsServiceInf service = **new**

BbsServiceImpl(); List<ForumEntity> forumList =

service.findForumList();

request.setAttribute("forumList", forumList);

request.getRequestDispatcher("forumlist.jsp")

.forward(request, response);

}

}

1. ServiceImpl

**public class** BbsServiceImpl **implements**

BbsServiceInf{

**public** List<ForumEntity> findForumList() { BbsDaoInf bbsDao = **new** BbsDaoImpl();

UserDaoInf userDao = **new** UserDaoImpl();

List<ForumEntity> forumList = bbsDao.findForumList();

**for** (ForumEntity forumEntity : forumList) {

**int** uid = forumEntity.getAuthor().getUser\_id();

UserEntity author = userDao.findUserById(uid);

forumEntity.setAuthor(author);

}

**return** forumList;

}

**public** ForumEntity findReplyList(**int** fid) { BbsDaoInf bbsDao = **new** BbsDaoImpl(); UserDaoInf userDao = **new** UserDaoImpl();

ForumEntity forum = bbsDao.findForumById(fid);

UserEntity author = userDao.findUserById(forum.getAuthor().getUser\_ id());

List<ReplyEntity> replyList = bbsDao.findReplyList(fid);

**for** (ReplyEntity replyEntity : replyList) { UserEntity replyUser =

userDao.findUserById(replyEntity.getUser().getU ser\_id());

replyEntity.setUser(replyUser);

}

forum.setAuthor(author); forum.setReplyList(replyList); **return** forum;

}

//查询个人发帖

**public** List<ForumEntity> personForum(**int** pid){ BbsDaoInf dao=**new** BbsDaoImpl(); List<ForumEntity> personList=**new**

ArrayList<ForumEntity>();

personList=dao.personForum(pid);

**return** (personList==**null**)?**null**:personList;

}

//查看单个用户发帖情况

**public** List<ForumEntity> seeUserForumList(**int** uid){

BbsDaoInf dao=**new** BbsDaoImpl(); List<ForumEntity>

forum=dao.personForum(uid);

**return** forum;

}

}

1. DaoImpl

**public class** BbsDaoImpl **implements** BbsDaoInf{

//查询帖子列表

**public** List<ForumEntity> findForumList() { Connection conn = **null**; PreparedStatement ps = **null**;

ResultSet rs = **null**; List<ForumEntity> list = **new**

ArrayList<ForumEntity>();

String sql = "select \* from forum\_info";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); rs = ps.executeQuery(); **while**(rs.next()){

ForumEntity forum = **new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4));

//获取对应用户表中的外键

UserEntity author = **new** UserEntity(); author.setUser\_id(rs.getInt(5)); forum.setAuthor(author); list.add(forum);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** list;

}

//根据ID查询某个帖子

**public** ForumEntity findForumById(**int** fid) { Connection conn = **null**;

PreparedStatement ps = **null**; ResultSet rs = **null**; ForumEntity forum = **null**;

String sql = "select \* from forum\_info where fid = ?";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setInt(1, fid);

rs = ps.executeQuery();

**if**(rs.next()){

forum = **new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4));

//获取对应用户表中的外键

UserEntity author = **new** UserEntity(); author.setUser\_id(rs.getInt(5)); forum.setAuthor(author);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** forum;

}

//查询帖子对应的所有回复

**public** List<ReplyEntity> findReplyList(**int** fid)

{

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs = **null**; List<ReplyEntity> list = **new**

ArrayList<ReplyEntity>();

//根据外键作为条件查询当前帖子的所有回复String sql = "select \* from reply\_info where

fid = ?";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setInt(1, fid);

rs = ps.executeQuery();

**while**(rs.next()){

ReplyEntity reply = **new** ReplyEntity(); reply.setRid(rs.getInt(1)); reply.setContent(rs.getString(2));

reply.setReplyTime(rs.getTimestamp(3));

//获取对应用户表中的外键UserEntity replyUser = **new**

UserEntity();

replyUser.setUser\_id(rs.getInt(4)); reply.setUser(replyUser); list.add(reply);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** list;

}

//根据用户ID查询用户的所有发帖

**public** List<ForumEntity> personForum(**int** pid){

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs=**null**;

String sql="select \* from forum\_info where user\_id=?";

List<ForumEntity> list=**new** ArrayList<ForumEntity>();

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, pid); rs=ps.executeQuery();

**while**(rs.next()){

ForumEntity forum=**new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4)); UserEntity user=**new** UserEntity(); user.setUser\_id(rs.getInt(5)); forum.setAuthor(user); list.add(forum);

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}

**return** list;

}

}

* + 1. 发帖

1. Servlet

@SuppressWarnings("serial")

**public class** ReleaseServlet **extends** HttpServlet {

**public void** service(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException { request.setCharacterEncoding("utf-8"); String title =

request.getParameter("title"); String content =

request.getParameter("content");

UserEntity user = (UserEntity)request.getSession().getAttribute(" user");

ForumEntity forum = **new** ForumEntity(); forum.setTitle(title); forum.setContent(content); forum.setAuthor(user); forum.setCreateTime(**new**

java.sql.Timestamp(**new** java.util.Date().getTime()));

BbsServiceInf service = **new**

BbsServiceImpl(); service.createForum(forum); response.sendRedirect("forumList");

}

}

1. Service

**public class** BbsServiceImpl **implements**

BbsServiceInf{

**public** List<ForumEntity> findForumList() { BbsDaoInf bbsDao = **new** BbsDaoImpl(); UserDaoInf userDao = **new** UserDaoImpl();

List<ForumEntity> forumList = bbsDao.findForumList();

**for** (ForumEntity forumEntity : forumList) {

**int** uid = forumEntity.getAuthor().getUser\_id();

UserEntity author = userDao.findUserById(uid);

forumEntity.setAuthor(author);

}

**return** forumList;

}

//发表帖子

**public int** createForum(ForumEntity forum) { BbsDaoInf bbsDao = **new** BbsDaoImpl(); **return** bbsDao.addForum(forum);

}

**public** ForumEntity findReplyList(**int** fid) { BbsDaoInf bbsDao = **new** BbsDaoImpl(); UserDaoInf userDao = **new** UserDaoImpl();

ForumEntity forum = bbsDao.findForumById(fid);

UserEntity author = userDao.findUserById(forum.getAuthor().getUser\_

id());

List<ReplyEntity> replyList = bbsDao.findReplyList(fid);

**for** (ReplyEntity replyEntity : replyList) { UserEntity replyUser =

userDao.findUserById(replyEntity.getUser().getU ser\_id());

replyEntity.setUser(replyUser);

}

forum.setAuthor(author); forum.setReplyList(replyList); **return** forum;

}

}

1. DaoImpl

**public class** BbsDaoImpl **implements** BbsDaoInf{

//查询帖子列表

**public** List<ForumEntity> findForumList() {

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs = **null**; List<ForumEntity> list = **new**

ArrayList<ForumEntity>();

String sql = "select \* from forum\_info";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); rs = ps.executeQuery(); **while**(rs.next()){

ForumEntity forum = **new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4));

//获取对应用户表中的外键

UserEntity author = **new** UserEntity(); author.setUser\_id(rs.getInt(5)); forum.setAuthor(author); list.add(forum);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** list;

}

//添加帖子

**public int** addForum(ForumEntity forum) { Connection conn = **null**; PreparedStatement ps = **null**;

String sql = "insert into forum\_info(title,content,create\_time,user\_id) values(?,?,?,?)";

System.*out*.println("添加帖子");

**int** i = 0;

**try**{

conn = DBUtil.*getConnection*();

ps = conn.prepareStatement(sql); ps.setString(1, forum.getTitle()); ps.setString(2, forum.getContent()); ps.setTimestamp(3,

forum.getCreateTime());

ps.setInt(4, forum.getAuthor().getUser\_id());

i = ps.executeUpdate();

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(**null**, ps, conn);

}

**return** i;

}

//根据ID查询某个帖子

**public** ForumEntity findForumById(**int** fid) { Connection conn = **null**;

PreparedStatement ps = **null**; ResultSet rs = **null**; ForumEntity forum = **null**;

String sql = "select \* from forum\_info where fid = ?";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setInt(1, fid);

rs = ps.executeQuery();

**if**(rs.next()){

forum = **new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4));

//获取对应用户表中的外键

UserEntity author = **new** UserEntity(); author.setUser\_id(rs.getInt(5)); forum.setAuthor(author);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** forum;

}

//查询帖子对应的所有回复

**public** List<ReplyEntity> findReplyList(**int** fid)

{

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs = **null**; List<ReplyEntity> list = **new**

ArrayList<ReplyEntity>();

//根据外键作为条件查询当前帖子的所有回复String sql = "select \* from reply\_info where

fid = ?";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setInt(1, fid);

rs = ps.executeQuery();

**while**(rs.next()){

ReplyEntity reply = **new** ReplyEntity(); reply.setRid(rs.getInt(1)); reply.setContent(rs.getString(2));

reply.setReplyTime(rs.getTimestamp(3));

//获取对应用户表中的外键UserEntity replyUser = **new**

UserEntity();

replyUser.setUser\_id(rs.getInt(4)); reply.setUser(replyUser); list.add(reply);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** list;

}

//根据用户ID查询用户的所有发帖

**public** List<ForumEntity> personForum(**int**

pid){

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs=**null**;

String sql="select \* from forum\_info where user\_id=?";

List<ForumEntity> list=**new** ArrayList<ForumEntity>();

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, pid); rs=ps.executeQuery();

**while**(rs.next()){

ForumEntity forum=**new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4)); UserEntity user=**new** UserEntity(); user.setUser\_id(rs.getInt(5)); forum.setAuthor(user); list.add(forum);

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}

**return** list;

}

//用户删除个人某个帖子

**public void** deleteForum(**int** fid){ Connection conn=**null**; PreparedStatement ps=**null**;

String sql="delete from forum\_info where

fid=?";

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, fid); ps.executeUpdate();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{

DBUtil.*close*(**null**, ps, conn);

}

}

//根据帖子id删除所有回复

**public void** deleteReply(**int** fid){

System.*out*.println(fid); Connection conn=**null**; PreparedStatement ps=**null**;

String sql="delete from reply\_info where

fid=?";

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, fid);

ps.executeUpdate();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{ DBUtil.*close*(**null**,ps,conn);

}

}

}

* + 1. 回帖

1. Servlet

@SuppressWarnings("serial")

**public class** ReplyServlet **extends** HttpServlet{

**public void** service(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException { request.setCharacterEncoding("utf-8"); **int** fid =

Integer.*parseInt*(request.getParameter("fid"));

**int** uid = Integer.*parseInt*(request.getParameter("uid"));

String replyContent = request.getParameter("content");

ForumEntity forum = **new** ForumEntity(); forum.setFid(fid);

UserEntity user = **new** UserEntity(); user.setUser\_id(uid);

ReplyEntity reply = **new** ReplyEntity(); reply.setForum(forum); reply.setUser(user); reply.setContent(replyContent); reply.setReplyTime(**new**

java.sql.Timestamp(**new** java.util.Date().getTime()));

BbsServiceInf service = **new**

BbsServiceImpl(); service.createReply(reply);

response.sendRedirect("replyList?fid="+fid);

}

}

1. Service

**public class** BbsServiceImpl **implements**

BbsServiceInf{

**public** List<ForumEntity> findForumList() { BbsDaoInf bbsDao = **new** BbsDaoImpl(); UserDaoInf userDao = **new** UserDaoImpl();

List<ForumEntity> forumList =

bbsDao.findForumList();

**for** (ForumEntity forumEntity : forumList) {

**int** uid = forumEntity.getAuthor().getUser\_id();

UserEntity author = userDao.findUserById(uid);

forumEntity.setAuthor(author);

}

**return** forumList;

}

**public** ForumEntity findReplyList(**int** fid) { BbsDaoInf bbsDao = **new** BbsDaoImpl(); UserDaoInf userDao = **new** UserDaoImpl();

ForumEntity forum = bbsDao.findForumById(fid);

UserEntity author = userDao.findUserById(forum.getAuthor().getUser\_ id());

List<ReplyEntity> replyList = bbsDao.findReplyList(fid);

**for** (ReplyEntity replyEntity : replyList) { UserEntity replyUser =

userDao.findUserById(replyEntity.getUser().getU ser\_id());

replyEntity.setUser(replyUser);

}

forum.setAuthor(author); forum.setReplyList(replyList); **return** forum;

}

//发表回复

**public int** createReply(ReplyEntity reply) { BbsDaoInf dao = **new** BbsDaoImpl();

**return** dao.addReply(reply);

}

}

1. DaoImpl

**public class** BbsDaoImpl **implements** BbsDaoInf{

//查询帖子对应的所有回复

**public** List<ReplyEntity> findReplyList(**int** fid)

{

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs = **null**; List<ReplyEntity> list = **new**

ArrayList<ReplyEntity>();

//根据外键作为条件查询当前帖子的所有回复String sql = "select \* from reply\_info where

fid = ?";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setInt(1, fid);

rs = ps.executeQuery();

**while**(rs.next()){

ReplyEntity reply = **new** ReplyEntity(); reply.setRid(rs.getInt(1)); reply.setContent(rs.getString(2));

reply.setReplyTime(rs.getTimestamp(3));

//获取对应用户表中的外键UserEntity replyUser = **new**

UserEntity();

replyUser.setUser\_id(rs.getInt(4)); reply.setUser(replyUser); list.add(reply);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** list;

}

//添加回复

**public int** addReply(ReplyEntity reply) {

Connection conn = **null**; PreparedStatement ps = **null**; **int** i = 0;

System.*out*.println("添加回复成功");

String sql = "insert into reply\_info(reply\_content,reply\_time,user\_id,fid) values(?,?,?,?)";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql);

ps.setString(1, reply.getContent()); ps.setTimestamp(2,

reply.getReplyTime());

//将相应的两个外键设值ps.setInt(3,

reply.getUser().getUser\_id());

ps.setInt(4, reply.getForum().getFid()); i = ps.executeUpdate();

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(**null**, ps, conn);

}

**return** i;

}

}

* + 1. 删帖

1. servlet

@SuppressWarnings("serial")

**public class** MdeleteUserForumServlet **extends**

HttpServlet {

**public void** service(HttpServletRequest request,HttpServletResponse response) **throws** ServletException,IOException{

String forumid=request.getParameter("fid");

**int** fid=Integer.*parseInt*(forumid); String uid=request.getParameter("uid");

BbsServiceInf service=**new** BbsServiceImpl();

service.deleteReplyByFid(fid); service.deleteForumByFid(fid);

request.setAttribute("uid", uid);

request.getRequestDispatcher("MSeeUserForum")

.forward(request, response);

}

}

1. service

**public class** BbsServiceImpl **implements**

BbsServiceInf{

**public** List<ForumEntity> findForumList() { BbsDaoInf bbsDao = **new** BbsDaoImpl(); UserDaoInf userDao = **new** UserDaoImpl();

List<ForumEntity> forumList = bbsDao.findForumList();

**for** (ForumEntity forumEntity : forumList) {

**int** uid = forumEntity.getAuthor().getUser\_id();

UserEntity author = userDao.findUserById(uid);

forumEntity.setAuthor(author);

}

**return** forumList;

}

**public** ForumEntity findReplyList(**int** fid) { BbsDaoInf bbsDao = **new** BbsDaoImpl(); UserDaoInf userDao = **new** UserDaoImpl();

ForumEntity forum = bbsDao.findForumById(fid);

UserEntity author = userDao.findUserById(forum.getAuthor().getUser\_ id());

List<ReplyEntity> replyList = bbsDao.findReplyList(fid);

**for** (ReplyEntity replyEntity : replyList) { UserEntity replyUser =

userDao.findUserById(replyEntity.getUser().getU ser\_id());

replyEntity.setUser(replyUser);

}

forum.setAuthor(author); forum.setReplyList(replyList); **return** forum;

}

//查询个人发帖

**public** List<ForumEntity> personForum(**int** pid){ BbsDaoInf dao=**new** BbsDaoImpl(); List<ForumEntity> personList=**new**

ArrayList<ForumEntity>();

personList=dao.personForum(pid);

**return** (personList==**null**)?**null**:personList;

}

//删除个人帖子

**public void** deleteForumByFid(**int** fid){ BbsDaoInf dao=**new** BbsDaoImpl(); dao.deleteForum(fid);

}

//删除帖子的所有回复

**public void** deleteReplyByFid(**int** fid){ BbsDaoInf dao=**new** BbsDaoImpl(); dao.deleteReply(fid);

}

//删除帖子下的某个用户的回帖

**public void** deleteReplyByRid(**int** rid){ BbsDaoInf dao=**new** BbsDaoImpl();

dao.deleteReplyByRid(rid);

}

//管理员模块

//删除用户（删除用户信息，发帖信息，回帖信息）

**public void** dUserAllForumByUid(**int** uid){ BbsDaoInf dao=**new** BbsDaoImpl(); dao.dUserAllforumByUid(uid);

}

//删除该用户所有回帖

**public void** dUserAllreplyByUid(**int** uid){ BbsDaoInf dao=**new** BbsDaoImpl(); dao.dUserAllreplyByUid(uid);

}

//删除用户帖子下的所有回帖

**public void** dUserOtherRelyByFid(**int** uid){ BbsDaoInf dao=**new** BbsDaoImpl(); dao.dUserOtherRelyByFid(uid);

}

//查看单个用户发帖情况

**public** List<ForumEntity> seeUserForumList(**int** uid){

BbsDaoInf dao=**new** BbsDaoImpl(); List<ForumEntity>

forum=dao.personForum(uid);

**return** forum;

}

}

1. DaoImpl

**public class** BbsDaoImpl **implements** BbsDaoInf{

//查询帖子列表

**public** List<ForumEntity> findForumList() { Connection conn = **null**; PreparedStatement ps = **null**;

ResultSet rs = **null**; List<ForumEntity> list = **new**

ArrayList<ForumEntity>();

String sql = "select \* from forum\_info";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

**while**(rs.next()){

ForumEntity forum = **new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4));

//获取对应用户表中的外键

UserEntity author = **new** UserEntity(); author.setUser\_id(rs.getInt(5)); forum.setAuthor(author); list.add(forum);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** list;

}

//根据ID查询某个帖子

**public** ForumEntity findForumById(**int** fid) { Connection conn = **null**;

PreparedStatement ps = **null**; ResultSet rs = **null**; ForumEntity forum = **null**;

String sql = "select \* from forum\_info where fid = ?";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setInt(1, fid);

rs = ps.executeQuery();

**if**(rs.next()){

forum = **new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4));

//获取对应用户表中的外键

UserEntity author = **new** UserEntity(); author.setUser\_id(rs.getInt(5)); forum.setAuthor(author);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** forum;

}

//查询帖子对应的所有回复

**public** List<ReplyEntity> findReplyList(**int** fid)

{

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs = **null**; List<ReplyEntity> list = **new**

ArrayList<ReplyEntity>();

//根据外键作为条件查询当前帖子的所有回复

String sql = "select \* from reply\_info where fid = ?";

**try**{

conn = DBUtil.*getConnection*(); ps = conn.prepareStatement(sql); ps.setInt(1, fid);

rs = ps.executeQuery();

**while**(rs.next()){

ReplyEntity reply = **new** ReplyEntity(); reply.setRid(rs.getInt(1)); reply.setContent(rs.getString(2));

reply.setReplyTime(rs.getTimestamp(3));

//获取对应用户表中的外键UserEntity replyUser = **new**

UserEntity();

replyUser.setUser\_id(rs.getInt(4)); reply.setUser(replyUser); list.add(reply);

}

}**catch**(Exception e){ e.printStackTrace();

}**finally**{

DBUtil.*close*(rs, ps, conn);

}

**return** list;

}

//根据用户ID查询用户的所有发帖

**public** List<ForumEntity> personForum(**int** pid){

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs=**null**;

String sql="select \* from forum\_info where user\_id=?";

List<ForumEntity> list=**new** ArrayList<ForumEntity>();

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql);

ps.setInt(1, pid); rs=ps.executeQuery();

**while**(rs.next()){

ForumEntity forum=**new** ForumEntity(); forum.setFid(rs.getInt(1)); forum.setTitle(rs.getString(2)); forum.setContent(rs.getString(3));

forum.setCreateTime(rs.getTimestamp(4)); UserEntity user=**new** UserEntity(); user.setUser\_id(rs.getInt(5)); forum.setAuthor(user); list.add(forum);

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}

**return** list;

}

//用户删除个人某个帖子

**public void** deleteForum(**int** fid){ Connection conn=**null**; PreparedStatement ps=**null**;

String sql="delete from forum\_info where

fid=?";

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, fid); ps.executeUpdate();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{

DBUtil.*close*(**null**, ps, conn);

}

}

//根据帖子id删除所有回复

**public void** deleteReply(**int** fid){

System.*out*.println(fid); Connection conn=**null**; PreparedStatement ps=**null**;

String sql="delete from reply\_info where

fid=?";

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, fid);

ps.executeUpdate();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{ DBUtil.*close*(**null**,ps,conn);

}

}

//用户删除某个回帖

**public void** deleteReplyByRid(**int** rid){

Connection conn=**null**; PreparedStatement ps=**null**;

String sql="delete from reply\_info where reply\_id=?";

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, rid);

ps.executeUpdate();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{ DBUtil.*close*(**null**,ps,conn);

}

}

//用户管理模块

//删除用户所有发帖

**public void** dUserAllforumByUid(**int** uid){ Connection conn=**null**; PreparedStatement ps=**null**;

String sql="delete from forum\_info where user\_id=?";

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, uid);

ps.executeUpdate();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{

DBUtil.*close*(**null**,ps,conn);

}

}

//删除用户的所有回帖

**public void** dUserAllreplyByUid(**int** uid){ Connection conn=**null**; PreparedStatement ps=**null**;

String sql="delete from reply\_info where user\_id=?";

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, uid);

ps.executeUpdate();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{

DBUtil.*close*(**null**,ps,conn);

}

}

//删除用户帖子下的所有回帖

**public void** dUserOtherRelyByFid(**int** uid){ Connection conn=**null**;

PreparedStatement ps=**null**; ResultSet rs=**null**;

String sql="select fid from forum\_info where user\_id=?";

List<ForumEntity> fidlist=**new** ArrayList<ForumEntity>();

**try** { conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(sql); ps.setInt(1, uid); rs=ps.executeQuery(); **while**(rs.next()){

ForumEntity forum=**new** ForumEntity();

forum.setFid(rs.getInt(1)); fidlist.add(forum);

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}**finally**{ DBUtil.*close*(**null**,ps,conn);

}

String fsql="delete from reply\_info where

fid=?";

# try {

**for**(ForumEntity forum : fidlist){ conn=DBUtil.*getConnection*(); ps=conn.prepareStatement(fsql); ps.setInt(1, forum.getFid());

ps.executeUpdate(); DBUtil.*close*(**null**,ps,conn);

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block e.printStackTrace();

}

}

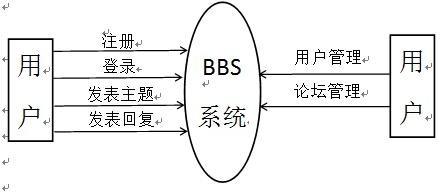
}

1. 数据库

数据流图

数据流图，简称 DFD，是 SA 方法中用于表示系统逻辑模型的一种工具，它以图形的方式描绘数据在系统中流动和处理的过程，由于它只反映系统必须完成的逻辑功能，所以它 是一种功能模型。

* 1. 顶层数据流图



* 1. **0** 层数据流图



* 1. 具体数据流图
     1. 登录系统



* + 1. 注册系统



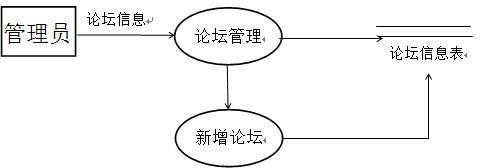
* + 1. 发表主题



* + 1. 回复主题

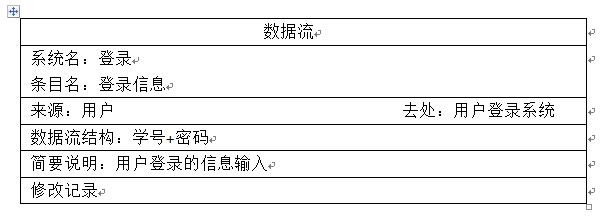


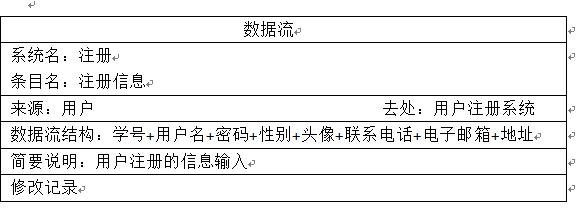
* + 1. 论坛管理



* 1. 数据字典
     1. 数据流

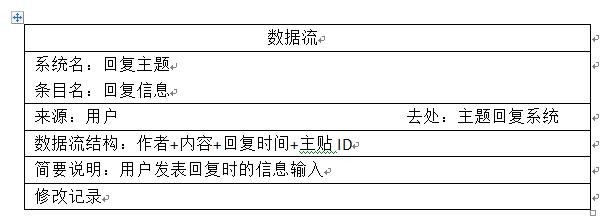
\*（1）



（2）

（3）









* + 1. 数据项

（1）



（2）



（3）



（4）

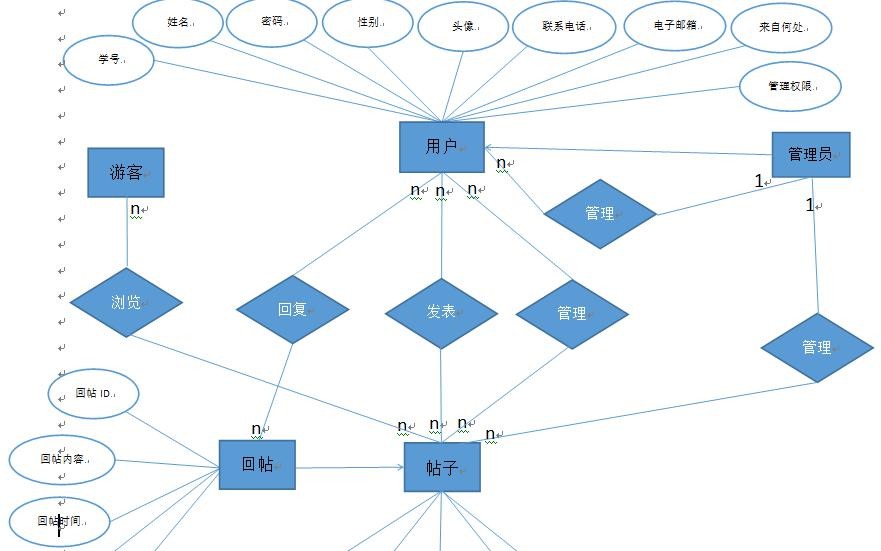
（5）



* 1. **ER** 图

E-R 图即实体-联系图(Entity Relationship Diagram)，是指提供了表示实体型、属性和联系的方法，用来描述现实世 界 的 概 念 模 型 。 E-R 方 法 : 是 “实 体 - 联 系 方 法 ”

（Entity-Relationship Approach）的简称。它是描述现实世界概念结构模型的有效方法。





* 1. 数据库设计
     1. 数据库分析

数据库的设计，在程序的开发中起着至关重要的作用， 它往往决定了在后面的开发中进行怎样的程序编码。一个合理、有限的数据库设计可降低程序的复杂性，使程序开发的过程更为容易。

本系统是一个中型的用户交流的网站，考虑到用户访问量以及网络连接，本系统采用 MySQL 作为数据库。

MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内， 这样就增加了速度并提高了灵活性。MySQL 的 SQL 语言是用于访问数据库的最常用标准化语言。

* + 1. 数据库的逻辑设计

根据系统需求，我们就可以创建系统所需要的数据库表

了。

本系统包含 3 个表，下面是这些表的结构。

* + - 1. user\_info 表结构如表所示：

user\_info 表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 字段长度 | 是否主键 | 是否为空 | 备注 |
| user\_id | int | 15 | 是 | 否 | 用户 id |
| user\_name | varchar | 50 | 否 | 否 | 用户名 |
| user\_password | varchar | 50 | 否 | 否 | 密码 |
| user\_sex | varchar | 2 | 否 | 否 | 性别 |
| user\_face | varchar | 255 | 否 | 否 | 头像 |
| user\_phone | varchar | 255 | 否 | 否 | 联系电话 |
| user\_email | varchar | 200 | 否 | 否 | 电子邮箱 |
| user\_from | varchar | 200 | 否 | 否 | 来自何处 |
| isAdmin | int | 2 | 否 | 是 | 是否为管理员 |

* + - 1. forum\_info 表结构如表所示：

forum\_info 表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 字段长度 | 是否主键 | 是否为空 | 备注 |
| Fid | int | 10 | 是 | 否 | 发帖 id |
| Title | varchar | 255 | 否 | 否 | 帖子标题 |
| content | varchar | 255 | 否 | 否 | 帖子内容 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| create\_time | datetime |  | 否 | 否 | 发帖时间 |
| user\_id | int | 11 | 否 | 否 | 用户 id |

* + - 1. reply\_info 表结构如表所示：

reply\_info 表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 字段长度 | 是否主键 | 是否为空 | 备注 |
| reply\_id | int | 10 | 是 | 否 | 回帖 id |
| reply\_content | varchar | 255 | 否 | 否 | 回帖内容 |
| reply\_time | datetime |  | 否 | 否 | 回帖时间 |
| user\_id | int | 11 | 否 | 否 | 用户 id |
| fid | int | 11 | 否 | 否 | 发帖 id |

* + 1. **SQL** 语句设计（建表语句 ）

（1）用户信息表（user\_info）

CREATE TABLE `user\_info` (

`user\_id` int(15) NOT NULL,

`user\_name` varchar(50) NOT NULL,

`user\_password` varchar(50) NOT NULL,

`user\_sex` varchar(2) NOT NULL,

`user\_face` varchar(255) NOT NULL,

`user\_phone` varchar(255) NOT NULL,

`user\_email` varchar(200) NOT NULL,

`user\_from` varchar(200) NOT NULL,

`isAdmin` int(2) DEFAULT NULL, PRIMARY KEY (`user\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

（2）主题信息表（forum\_info）

CREATE TABLE `forum\_info` (

`fid` int(10) NOT NULL AUTO\_INCREMENT,

`title` varchar(255) NOT NULL,

`content` varchar(255) NOT NULL,

`create\_time` datetime NOT NULL,

`user\_id` int(11) NOT NULL, PRIMARY KEY (`fid`)

) ENGINE=InnoDB AUTO\_INCREMENT=55 DEFAULT CHARSET=utf8

（3）回帖信息表（reply\_info）

CREATE TABLE `reply\_info` (

`reply\_id` int(10) NOT NULL AUTO\_INCREMENT,

`reply\_content` varchar(255) NOT NULL,

`reply\_time` datetime NOT NULL,

`user\_id` int(11) NOT NULL,

`fid` int(11) NOT NULL, PRIMARY KEY (`reply\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=52 DEFAULT CHARSET=utf8

**7.7MD5** 算法加密

（**1**）实现代码：

**public class** MD5 {

**public static** String generateCode(String str)

**throws** Exception{

MessageDigest md5=MessageDigest.*getInstance*("MD5");

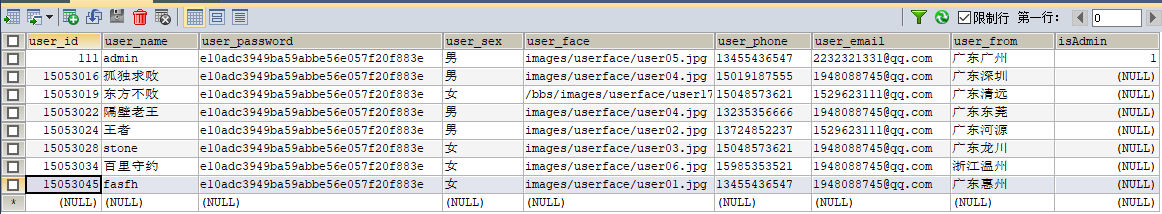
**byte**[] srcBytes=str.getBytes(); md5.update(srcBytes);

**byte**[] resultBytes=md5.digest(); String result=**new** String(resultBytes); **return** result;

}

}

（**2**）实现结果



1. 需求新增
   1. 初始需求

注册界面、登录界面、管理员模式、会员模式和游客模

式。

注册界面：

帮助游客注册，使游客获得会员身份，行使会员权利（也要履行会员义务），注册会员也有利于管理员管理会员的信息，有效监管会员的帖子，避免不法分子利用会员身份，在 论坛里散播不当内容。

管理员模式：

从后台管理会员的信息，包括管理会员的帖子和会员的留言，可以根据论坛及相关法律删除会员的不当内容，限制会员的使用权限

会员模式：

可以编辑自己的帖子，可以发布自己的帖子，可以删除自己的帖子，可以删除其他会员在自己帖子下的留言，可以游客模式：

可以浏览论坛主页，可以浏览注册界面和协议书，可以

浏览帖子

* 1. 第一次新增需求

第一次新增需求包括：

优化网页界面，新增管理员权限，新增会员帖子编辑功

能

新增管理员权限：

所有用户信息均可查阅，并且在用户违规发帖时有权限删除违规帖子和用户。

* 1. 第二次新增需求

第二次新增需求包括： 添加广告轮播。

广告轮播功能：

可以在轮播里放置宣传或者广告的图片。

9.功能确认

* 1. 测试目的

测试校园论坛各功能是否运行正常。

* 1. 测试人员
  2. 测试环境

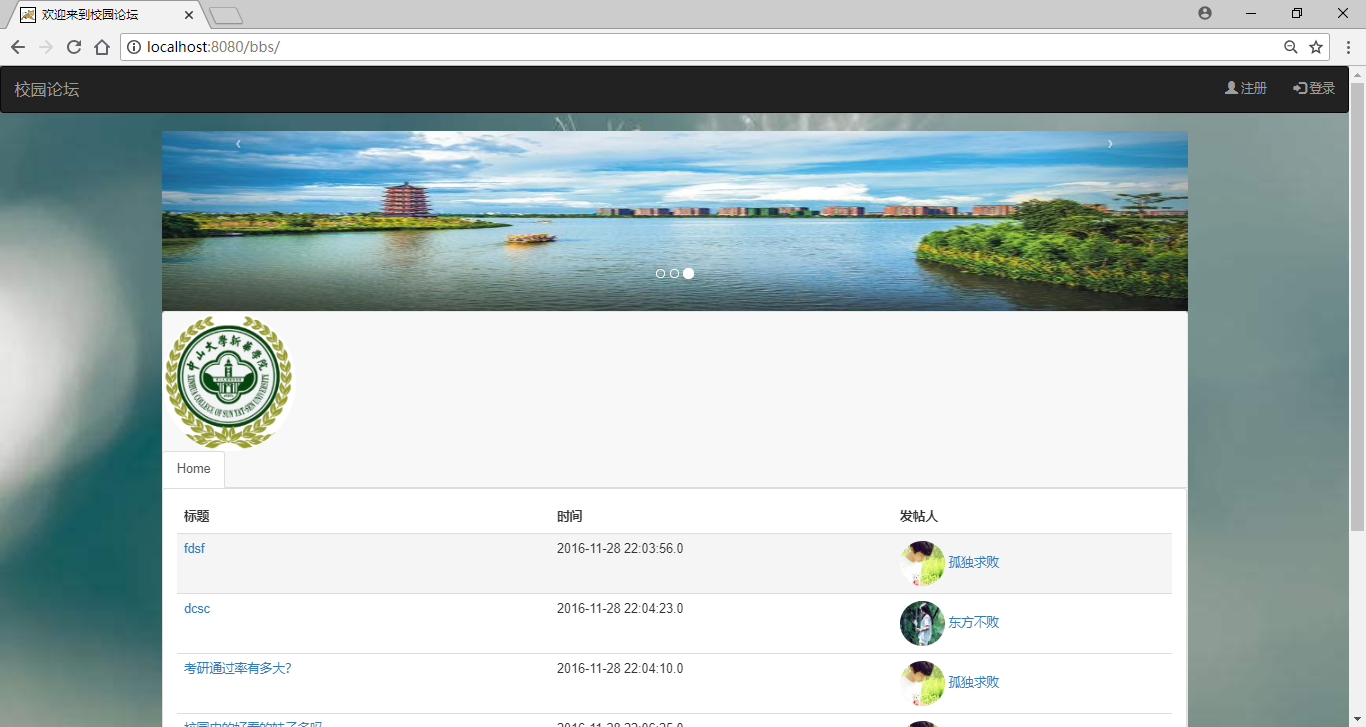
操作系统：windows10 浏览器：Google Chrome

* 1. 测试内容

|  |  |  |
| --- | --- | --- |
| 用户注册 | 输入学号、密码及个人信息，并且通过验  证码认证后，完成注册 | 是 |
| 用户登录 | 输入学号及密码完成登录 | 是 |
| 浏览帖子 | 不论是否登录，都可以正常浏览已有的帖  子 | 是 |
| 发帖 | 只有登录后，方可发帖 | 是 |
| 回复 | 只有登录后，方可回复 | 是 |
| 个人信息 | 查看个人信息（头像、学号、姓名、性别  联系电话、电子邮箱、来自哪里） | 、 是 |
| 修改个人信息 | 对个人信息（头像、学号、姓名、性别、联系电话、电子邮箱、来自哪里）进行修  改 | 是 |
| 退出登录 | 退出已登录状态 | 是 |

* 1. 功能截图

（1）校园论坛主页



在校园论坛主页，可作为游客身份浏览帖子，但只有注册、 登录之后，方可回复跟帖。主页提供直观的注册、登录按钮。

（2）用户注册

输入学号、用户名、密码，以及其它个人信息之后，即可

完成注册。并且为了网站安全，用户需要通过验证码验证。

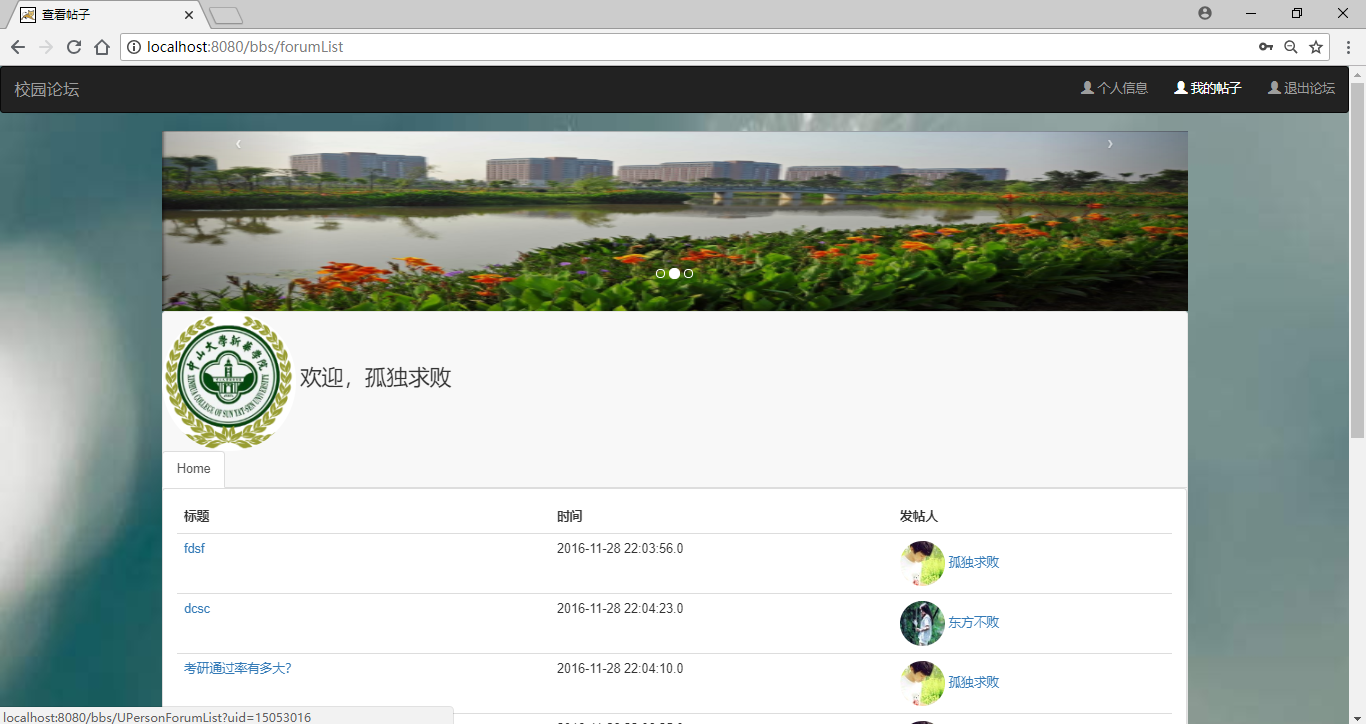
1. 用户登录

输入学号、密码即可完成登录。



若勾选“记住密码”，会自动填充学号及密码，方便用户快捷登录校园论坛。

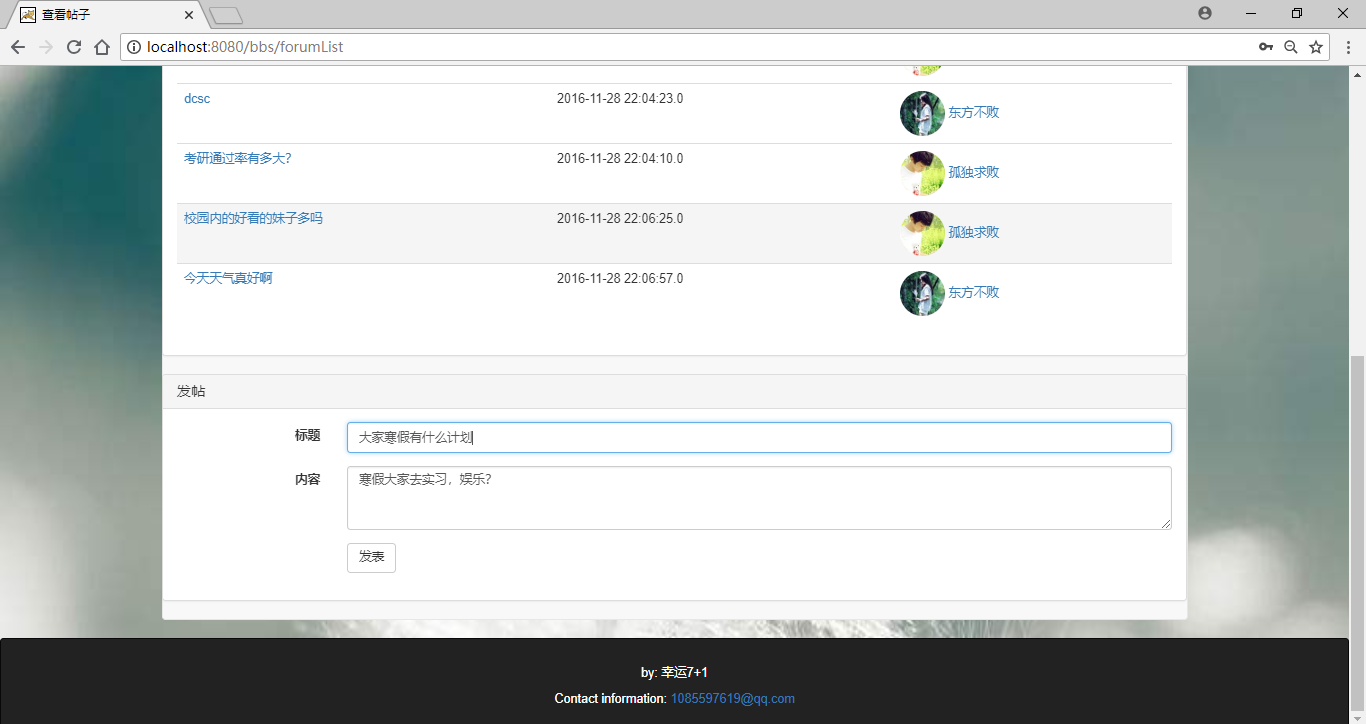
（4）查看帖子



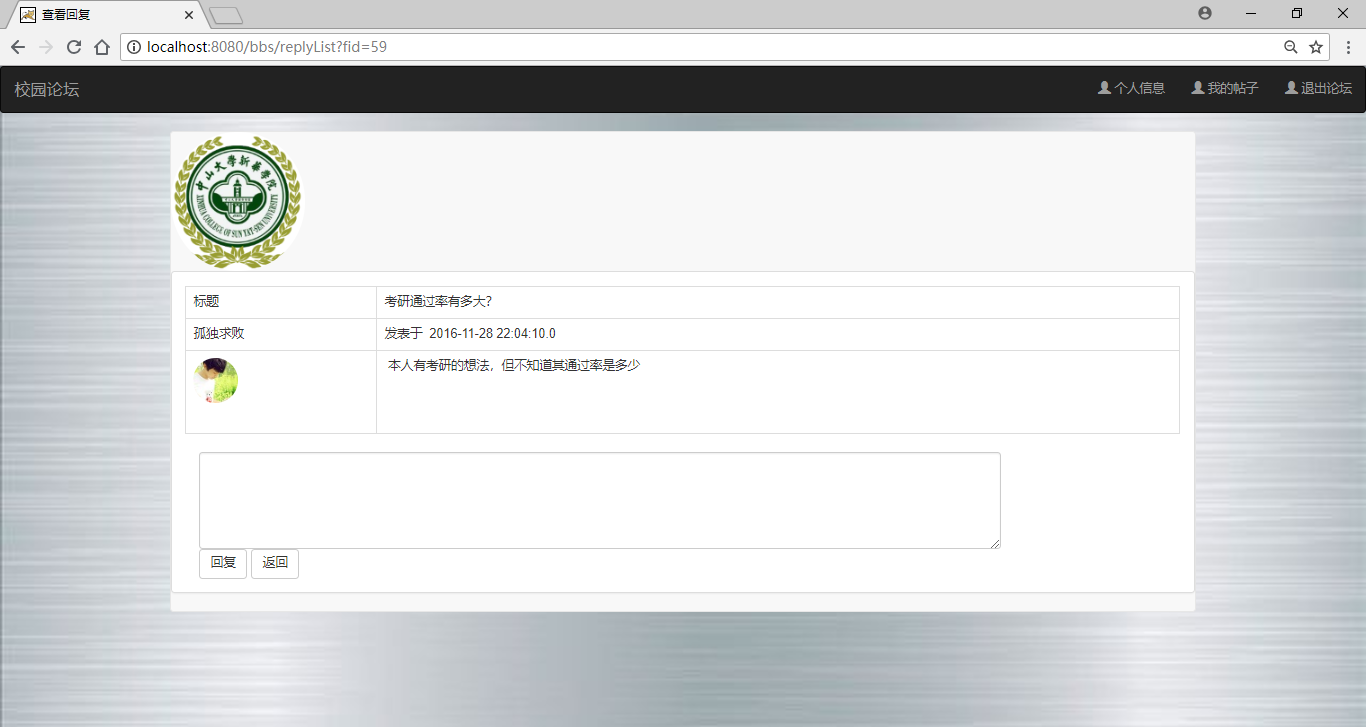
登录之后，进入“查看帖子”页面，可浏览已发布的帖子。

右上角提供“个人信息”、“ 我的帖子”、“退出论坛”三个按钮。

（5）发帖

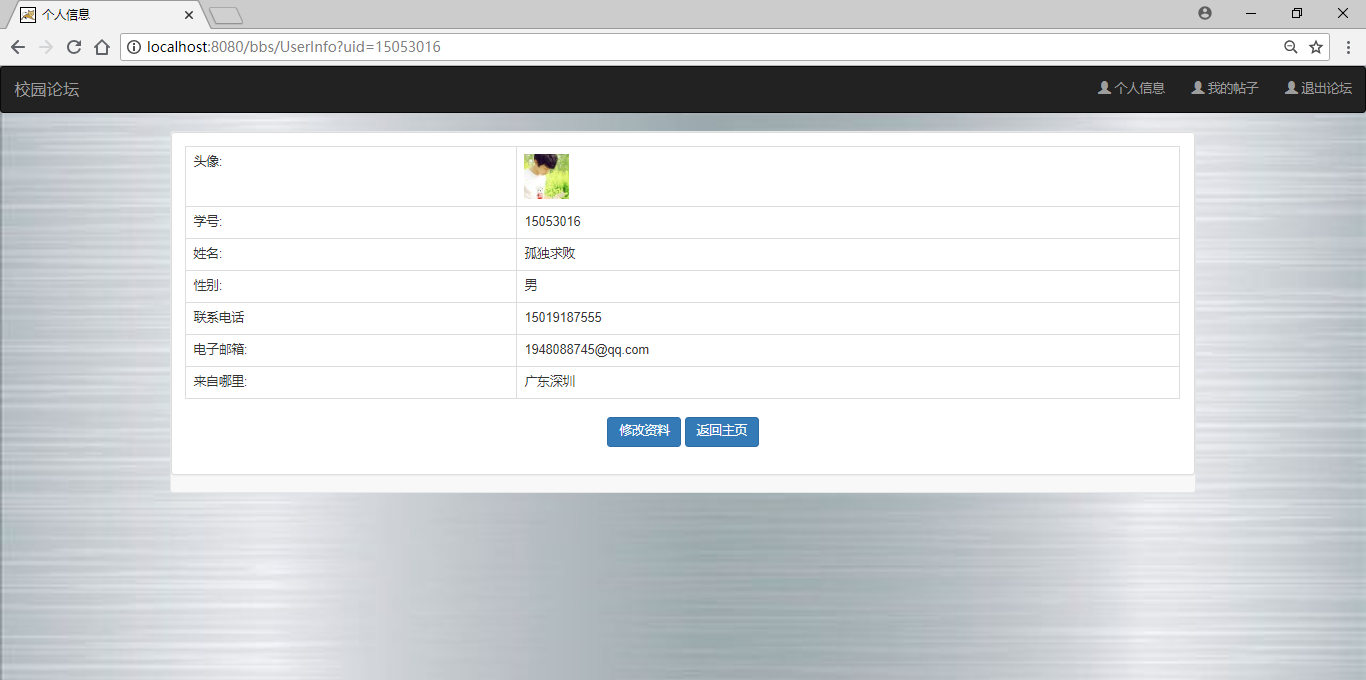
“查看帖子”页面底部，输入标题及内容，点击”发表”，即可发布自己的帖子。

（6）阅读、回复帖子



点击帖子的标题，即可阅读帖子详情，可以回复跟帖。

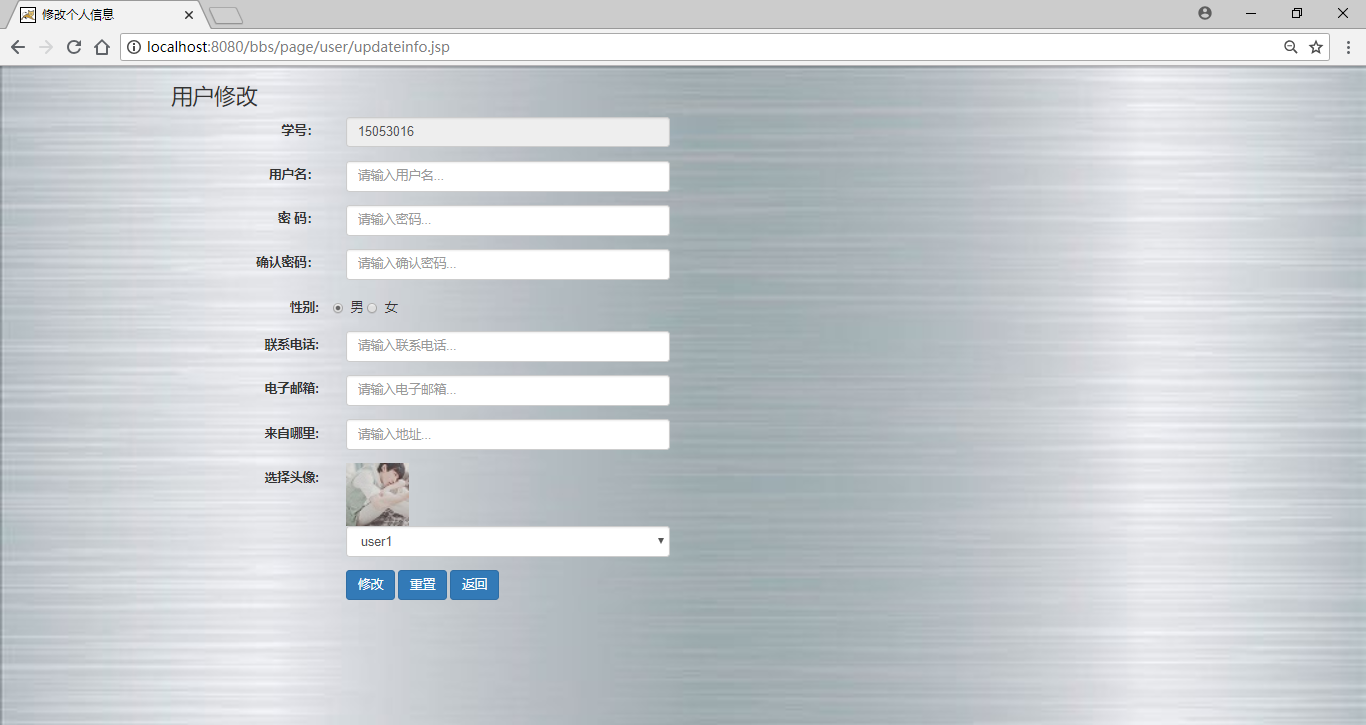
（7）个人信息



点击右上角的“个人信息”按钮，即可查看个人信息，包 括头像、学号、姓名、性别、联系电话、电子邮箱，及来自

哪里。

（8）修改个人信息



“个人信息”页面，点击“修改资料”按钮，即可进入“修改个人信息”页面。可对个人信息进行修改。

（9）退出登录



点击右上角的“退出论坛”按钮，即退出登录状态，回到

论坛主页。

* 1. 测试结论

校园论坛各功能都运行正常，运行状态良好，功能确认通 过。

1. 逻辑确认

表示层：负责界面和交互；

业务逻辑层：负责定义业务逻辑（规则、工作流、数据完整性等），接收来自表示层的数据请求，逻辑判断后，向数据访问层提交请求，并传递数据访问结果，业务逻辑层实际上是一个中间件，起着承上启下的重要作用；

数据访问层：负责数据读取。

* 1. 表示层
     1. **MVC** 中的视图层的设计

一个典型的 Web 应用的前端应该是表示层表示层由 UI 和 UI 控制逻辑组成。

# UI（User Interface）

UI 是客户端的用户界面，负责从用户方接收命令，请求，数

据，传递给业务层处理，然后将结果呈现出来。

1. **UI** 控制逻辑

UI 控制逻辑负责处理 UI 和业务层之间的数据交互，UI 之间状态流程的控制，同时负责简单的数据验证和格式化等功能。 主要功能如下：

管理用户的请求，做出相应的响应

提供一个流程控制器，委派调用业务逻辑和其他上层处理处理异常

为显示提供一个数据模型用户界面的验证

* + 1. 客户端的 **JavaScript** 脚本的应用

（**1**）应用 **JavaScript** 时的要点

在 MVC 的应用中，一般是不再鼓励使用客户端的校验技术， 而改用服务器端的校验技术以提高项目的安全性和更好的兼容性。

（**2**）**JavaScript** 在本项目中的应用之一

为了能够在客户端对用户的表单进行及时地检查和实现更 友好的人机交互，在某些页面中仍然使用客户端的 JavaScript 脚本技术----主要承担实时交互和实现动画效果。

* + 1. **Web** 系统安全性的考虑

本系统中的基本的安全性从以下几个方面加以实现

（1）注册验证码

利用一个后台服务器端的功能组件实现随机产生一个验证 码，由于验证码是机器随机产生的，因此暴力破解程序无法 预料到具体是什么，所以可以防止暴力破解。

（2）密码采用 MD5 加密

采用 MD5 算法对密码进行加密。

MD5 具 有 以 下 特 点1、压缩性：任意长度的数据，算出的 MD5 值长度都是固定的。

2、容易计算：从原数据计算出 MD5 值很容易。

3、抗修改性：对原数据进行任何改动，哪怕只修改 1 个字节，所得到的 MD5 值都有很大区别。

4、强抗碰撞：已知原数据和其 MD5 值，想找到一个具有相同 MD5 值的数据（即伪造数据）是非常困难的。

**10.2** 业务逻辑层

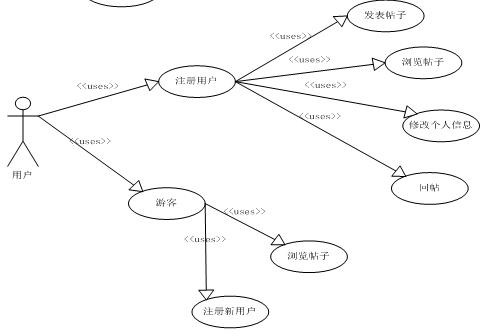
领域实体： 游客，论坛用户，评论，帖子，发帖，看帖

业务规则：游客注册成为论坛用户，论坛用户登陆看帖可评论，或发贴生成帖子，以供用户之间的交流。且游客只能注 册成为论坛用户或看帖。以此为基础更有其它扩展功能。

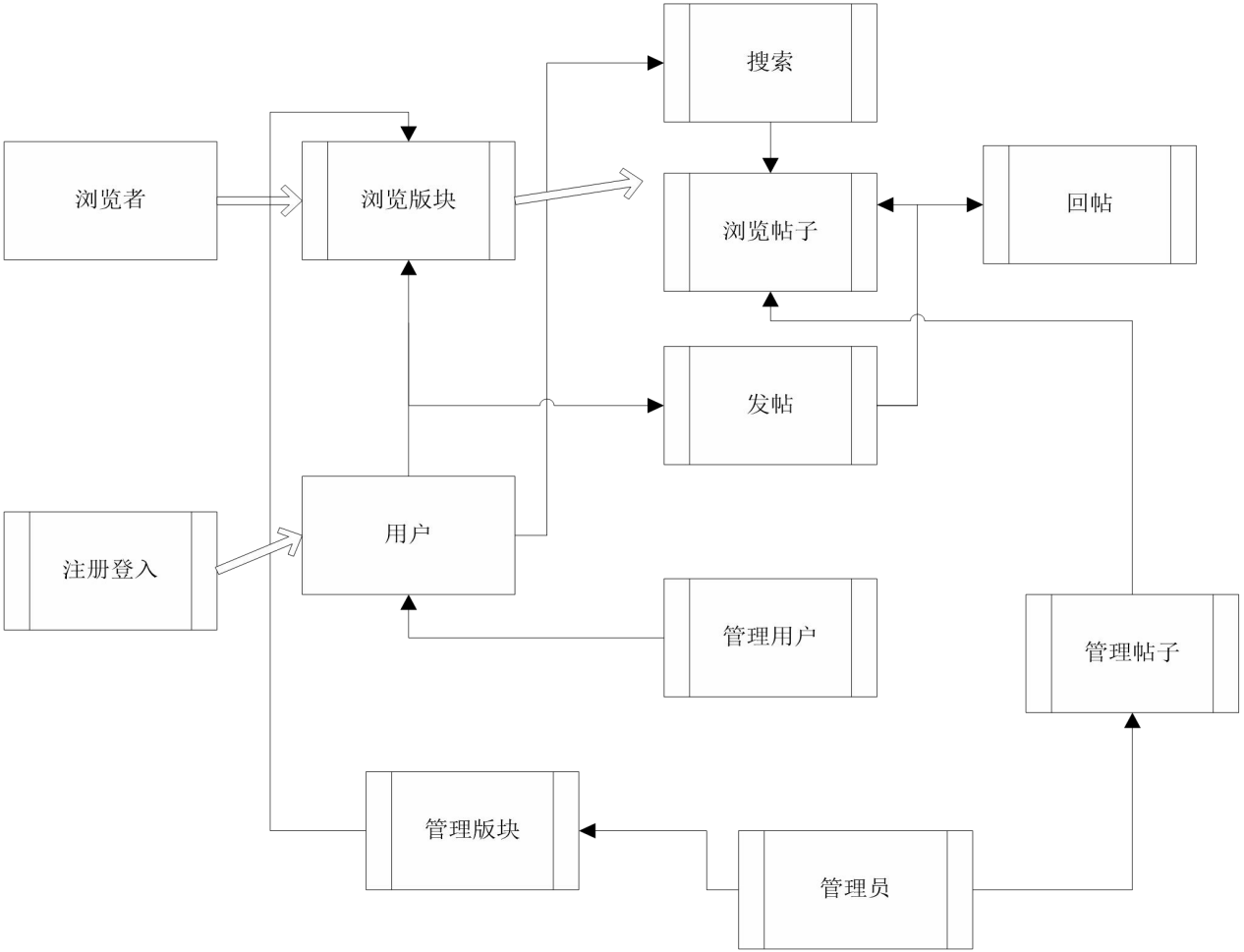
数据完整性：论坛用户评论发帖必须登陆账号，没有则不能

成功。

工作流：论坛用户-登陆-看帖-评论 论坛用户-发帖 游客- 注册 游客-看帖

如图所示：

数据访问：



1. 验收报告

|  |  |  |
| --- | --- | --- |
| 职务 | 姓名 | 日期 |
| 前端开发人员 |  | 2016.11.28 |
| 软件测试人员 |  | 2016.11.28 |
| 需求分析人员 |  | 2016.11.28 |
| 详细设计人员 |  | 2016.11.28 |
| 总体设计人员 |  | 2016.11.28 |
| 数据库管理人员 |  | 2016.11.28 |
| 后端开发人员 |  | 2016.11.28 |

* 1. 项目基本情况

|  |  |
| --- | --- |
| 项目名称 | 校园论坛 |
| 开发小组 |  |
| 项目开始时间 | 2016.09.12 |
| 项目验收时间 | 2016.11.27 |

* 1. 项目概述

开发校园论坛系统的目的是提供一个供我校学生交流的平台，为我校学生提供交流经验、探讨问题的社区。因此， 校园论坛系统最基本的功能首先是发表主题，其次是其他人员根据主题发表自己的看法。此外，为了记录主题的发表者和主题的回复者信息，系统还需要提供用户注册和登录的功能。只有注册的用户登录后才能够发表和回复主题，浏览者

（游客）只能浏览主题信息。根据用户的需求及以上的分析， 校园论坛需要具备前台功能和后台功能。

* 1. 验收测试环境
     1. 硬件

电脑：操作系统 Windows 7 及以上；有足够内存。网络。

交换机。

* + 1. 软件

Web 服务器：Tomcat 7.0

集成开发环境(IDE):Eclipse

数据库：MySQL 5.5

* + 1. 文档

1. 项目组成员任书
2. 需求分析报告
3. 需求变更
4. 项目周报
5. 操作指引
6. FS 业务说明书

g.数据库结构

h.需求新增

i. 功能确认

j. 逻辑确认

k. 验收报告

* + 1. 人员

前端开发人员：

需求分析人员：

总体设计人员：

详细设计人员:

数据库：

后端开发人员：

软件测试人员：

* 1. 验收及测试结果
     1. 产品及验收结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 产品项 | 验收结果 | 验收人 | 验收时间 |
|  |  | 接收/不接收 |  |  |
| 1 | 校园论坛 | 接收 |  | 2016.11.28 |
| 2 | 需求设计 | 接收 |  | 2016.11.28 |
| 3 | 总体设计 | 接收 |  | 2016.11.28 |
| 4 | 详细设计 | 接收 |  | 2016.11.28 |

* + 1. 产品功能验收结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能编号 | 功能名称 | 验收结果 | 验收人 | 验收时间 |
|  |  | 通过/不通过 |  |  |
| 001 | MD5加密 | 通过 |  | 2016.11.08 |
| 002 | 界面优化 | 通过 |  | 2016.11.15 |
| 003 | 登录 | 通过 |  | 2016.11.01 |
| 004 | 注册 | 通过 |  | 2016.11.01 |
| 005 | 主页轮播 | 通过 |  | 2016.11.16 |
| 006 | 修改资料 | 通过 |  | 2016.11.15 |
| 007 | 发表帖子 | 通过 |  | 2016.11.17 |
| 008 | 删除帖子 | 通过 |  | 2016.11.17 |
| 009 | 退出登录 | 通过 |  | 2016.11.18 |
| 010 | 注册时重置所填写的信息 | 通过 |  | 2016.11.17 |

* 1. 验收总结

项目验收小组一致认为：系统运行稳定，计算数据准确， 实现了最初确定的目标：

同时，项目验收小组一致认为，校园论坛项目是卓有成效的。把对软件系统的理解与对企业管理的深刻认识有机的结合起来，并应用到整个实施过程中。通过规范基础管理、统一名称和编码的系统应用准则和规程。

综合以上各方面因素，项目验收小组认为校园论坛系统实施达到了预期效果，符合基本要求。

* 1. 参考资料

|  |  |
| --- | --- |
| 序号 | 名称 |
| 1 | Java Web程序设计（第2版）（清华大学出版社） |
| 2 | 软件工程（第3版）（高等教育出版社） |
| 3 | 软件工程导论（第6版）（清华大学出版社） |
| 4 | 基于面向对象的软件分析与设计实验教程（中山大学出版社） |
| 5 | UML基础与建模实践教程（机械工业出版社） |
| 6 | 软件测试（第2版）（人民邮电出版社） |
| 7 | 数据库系统概论（第5版）（高等教育出版社） |