****

软件开发实训

说明书

题 目： **留言板系统设计**

学 院： 计算机与信息安全学院

专 业： 计算机科学与技术

学生姓名： 李义生

学 号： 1400310217

指导教师： 梁 海

2017年12月25日

## 1软件总体设计

本次软件开发实训所做的是一个留言板，通过前后端分离，实现用户能够使用封装的接口进行各种程序的开发，真正实现前后端的完全分离。本留言板分为两个部分，一部分是前端网页，实现一个小型的网页留言板；另一部分是后端，搭建一个服务器，编写后端服务程序将对数据库中各种操作进行封装，封装一个一个的接口，前端服务程序调用接口进行各种各种操作。

本软件的基本功能：注册、登录、注销、匿名提交留言、实名提交留言、留言管理（回复、删除）。留言只有在管理员回复以后才能显示在页面上。

基本流程如下：

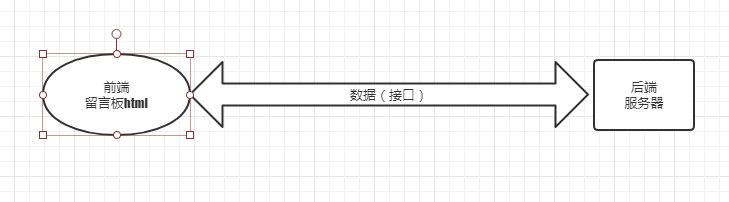


图1-1

## 2前端设计

本留言板前端是两个网页，分别为主网页和管理员网页;

2.1主网页

主网页分为分为三个部分：分别为系统菜单栏、登录栏、留言显示栏

系统菜单栏中包括三个按钮：刷新按钮、发表留言按钮、注销按钮；

刷新按钮：

点击刷新按钮网页自动刷新网页；

发留言按钮：

点击发留言按钮自动跳出一个文本框，用户可以在文本框中输入要提交的留言，点击发送，前端将数据封装后，通过服务器提供的接口将数据存储到服务器连接的数据库中，并将该文本框隐藏；如果数据输入栏中用户未填写任何的数据，系统在前端对数据进行判断是否为空，并提示不能为空，并跳转到主页面中。

点击注销按钮：

当点击注销按钮系统会自动退回到其实界面中。

登录栏：

登录栏中有四个部分分别为：主题文本框、管理员登录按钮、用户登录按钮、注册按钮。

主题文本框：

主图文本框中显示本系统的名称为messageboard 文本框和share your experiences文本框。

管理员登录按钮：

当用户点击管理员登录按钮时系统将本网页进行虚化，通过js弹出登录框，用户可以使用该登录框输入用户名和密码前端对数据进行封装后，通过登录接口传输到后台服务其中，等待后台服务程序对传入的的数据进行处理，返回一个结果，当数据库匹配成功后，跳转到管理员网页中表示用户登录成功。

用户登录按钮：

当用户点击登录按钮时系统将本网页进行虚化，通过js弹出登录框，用户可以使用该登录框输入用户名和密码前端对数据进行封装后，通过登录接口传输到后台服务其中，等待后台服务程序对传入的的数据进行处理，返回一个结果，当返回数据库匹配成功后。将该登录栏覆盖显示登录者的姓名。表示用户登录成功。

登录按钮实现效果如下图：

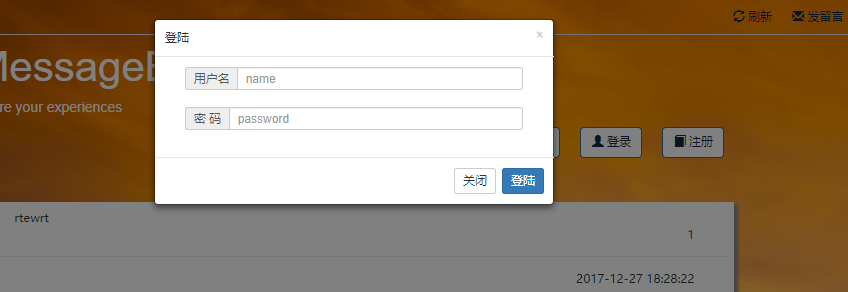


图2-1

注册按钮：

用用户点击注册按钮时系统将本网页进行虚化，，通过js弹出注册框，用户可以在注册框中输入名称和密码，前端调用对应的接口向后台发送数据，后台对数据进行处理后，与数据库中的数据进行匹配当，数据库中已存在该用户需要注册的数据时，返回该用户，如果不存在则返回空值，当返回的值为空值是则前台进行判断提示该用户已存在，否者注册成功。

注册按钮实现如下图：



图2-2

留言信息显示栏：

当网启动后自动调用调用数据接口，后台服务程序通过该接口发送所用已被管理员审核后的留言信息，前端对数据进行时间先后排序后将该每一条留言进行加载，在留言信息栏中显示每一条数据。并将每一条留言信息的时间和留言人的名字显示，如果后台通过接口返回的每一条信息的留言者的数据为空时，会自动显示为匿名。

主网页实现如图所示：

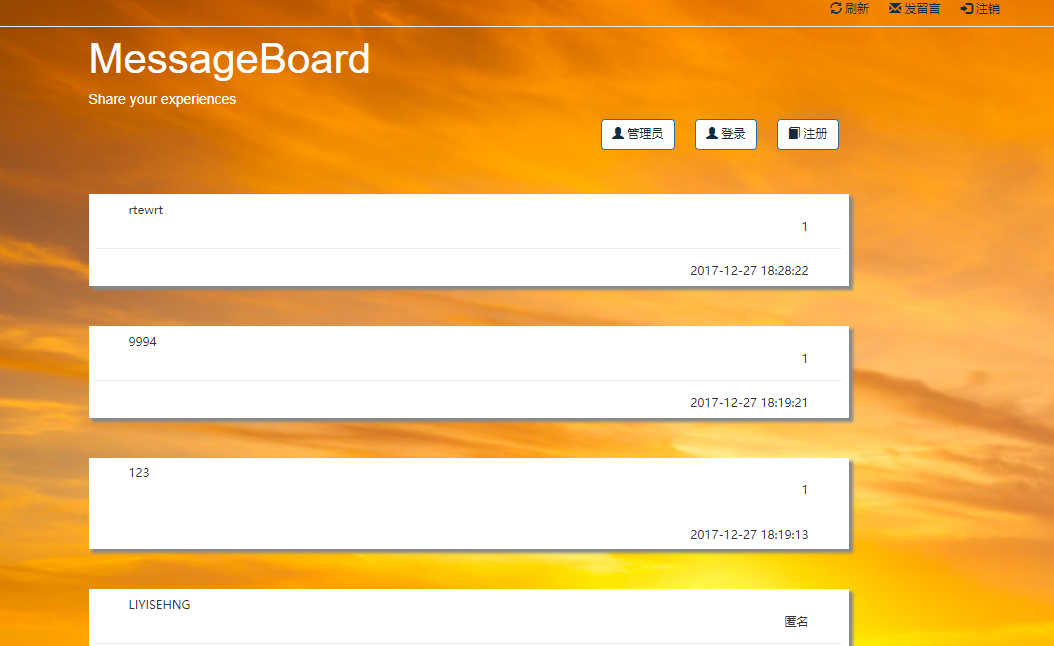


图2-3

2.2管理员网页

管理员网页包含三个部分：分别为系统菜单栏、网页信息栏、留言显示栏

系统菜单栏：

系统菜单栏中包括三个按钮：刷新按钮、发表留言按钮、注销按钮；

刷新按钮：

点击刷新按钮网页自动刷新网页；

注销按钮：

当用户点击注销按钮时退出管理员界面返回到主界面上，管理员的在该网页的登录信息也被注销，管理员需要进行登录才能返回到该界面上；

网页信息栏：

网页信息栏中显示本网页的名称“messageboard”文本和“你好：管理员”文本。

信息显示栏：

信息显示栏中主要显示管理员还未审核的留言信息，以及对每一条留言信息进行的操作按钮，分别为删除按钮和通过按钮。当管理员在主网页登录成功跳转到该网页的时候前台服务程序会直通调用获取信息接口，当后台系统接收到该接口请求时，后台服务程序会对该接口进行相应，并返回相应的数据，后台会进行数据库的查找并返回留言信息中没有被审核的数据，并将获取的数据通过该接口传输到前端，前端将该数据逐条显示在信息显示栏中，并在每一条留言信息上添加删除和通过按钮。

请求返回未审核的留言代码如下：

**function** shownorespondMessage(){

$.ajax({

url:"rest/message/getnorespondmessage",

type:"get",

success:**function**(data){

$("#showList").empty();

**var** html = $("#list").render(data.messagetable);

$("#showList").append(html);

}

})

}

点击每一条留言信息的删除按钮时，调用删除留言接口，后端服务器执行相应的操作，并将该条留言信息从数据库上删除。

该请求执行代码为：

function delelterespond(id){

$.ajax({

url:"rest/message/delete",

type:"get",

data:{b:id},

success:function(data){

//alert(JSON.stringify(data));

$("#showList").empty();

var html = $("#list").render(data.messagetable);

$("#showList").append(html);

}

});

}

当点击每一条留言信息的通过按钮时，调用信息通过接口，后端服务程序执行相应的操作，将数据库中该条数据的确认标志位做出相应的更改。

该请求接口的实现为：

function okrespond(id){

$.ajax({

url:"rest/message/respond",

type:"get",

data:{a:id},

success:function(data){

//alert(JSON.stringify(data));

$("#showList").empty();

var html = $("#list").render(data.messagetable);

$("#showList").append(html);

}

});

}

管理员界面图下图所示：



图 2-4

通过各个模块的搭配实现管理员界面。

## 3后台服务器设计

3.1数据库设计

本课程设计使用Mysql数据库，建立的数据库的名称为message，数据库中简历两个表格，分别为denglutable、messagetable。表denglutable中存储的是登录需要的信息。

表messagetable存储的是留言的信息。

Denglutable：

该表设计为四列，第一列为账号信息的编号主键id，第二列为账号信息的名字name，第三列为账号信息的密码password，第四列为账号信息的权限标志status，管理员权限为1，普通用户权限为0。

表denglutable如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| Id | name | psssword | status |
| 1 | 123456 | 123456 | 0 |
| 2 | 3 | 3 | 0 |
| 3 | 4 | 4 | 0 |
| 4 | 1 | 1 | 1 |
| 5 | 2 | 2 | 0 |

表3-1

messagetbale：

该表设计为五列，第一列为留言信息的编号主键id，第二列为该条留言的留言者的姓名name，第三列为该留言的内容message，第四列为该留言的留言时间，第五列为该留言有没有被审核的标志位respond，该标志为0为该条留言未被管理员审核，该标志位为1时表示该条留言已被管理员审核。

表messagetable如下表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | name | message | time | resond |
| 18 | 匿名 | sgfdg | 2017-12-27 22:07:02 | 0 |
| 19 | 匿名 | fsdgfsdg | 2017-12-27 22:07:08 | 0 |
| 20 | 匿名 | fgsdfgdsfg | 2017-12-27 22:07:13 | 0 |
| 21 | 123456 | 你好啊 | 2017-12-27 22:07:45 | 1 |

表3-2

3.2后端接口设计

后端分别写了两个接口类，分别为message类和dangluoperator类。message中是提供对用户登录，注册，管理员登录的接口方法的实现。message中提供的是对数据进行操作的一些接口方法的实现。

Dangluoperator的地址为(/user),其中包含的接口方法有:

登录接口（/denglu），该接口能够将服务端发送来请求，并将数据进行封装，将分装好的登录信息与数据库进行匹配，当数据库用户的信息符合是返回该用户，如果数据库中没有返回null。

管理员登录接口（/guanlidenglu）,该接口能够将服务端发送来请求，并将数据进行封装，将分装好的登录信息与数据库进行匹配，当数据库用户的信息符合是返回该用户，如果数据库中没有返回null，该接口主要为管理员登录而设计，其中添加了登录信息权限的验证，登录权限status为1时才能返回正确的结果。

注册接口(/zhuche)，该接口能够将前台发送的请求进行解析，并将用户的信息进行封装，匹配数据库中舒服存在该用户信息，如果没有提交到数据库中，并返回该用户。如果已存在则返回null。

Message的地址为（/sendmessage）,其中包含接口方法有：

留言接口（/sendmessage）,该接口能够将获得的数据进行封装，并自动生成系统时间，作为留言提交的时间，并存储到数据库中。

获取已审核留言接口（/getmessage），该接口能够从数据库中获取消息审核标志位respond为1的数据，并按照时间进行排序，并存储到消息对象的链表中，并将传输多条留言。

获取未审核留言（/getnorespondmessage），接口能够从数据库中获取消息审核标志位respond为0的数据，并按照时间进行排序，并存储到消息对象的链表中，并将传输多条留言。

管理员审核接口（/respond），该接口通过某一条消息的id，并将数据库中该条消息等审核标志位该置为1，同时向前端返回所有为被审核的信息。

删除留言（/delete），该接口通过将某一条消息的id，将该条消息从数据库中删除，并将所有未被审核的消息返回到前端。

各个接口的代码，请参考附录。

## 4功能测试

发送消息测试：

不登录发送一条消息：

例如：

“帅气的老师”

“我爱你”

登录管理员界面进行审核：

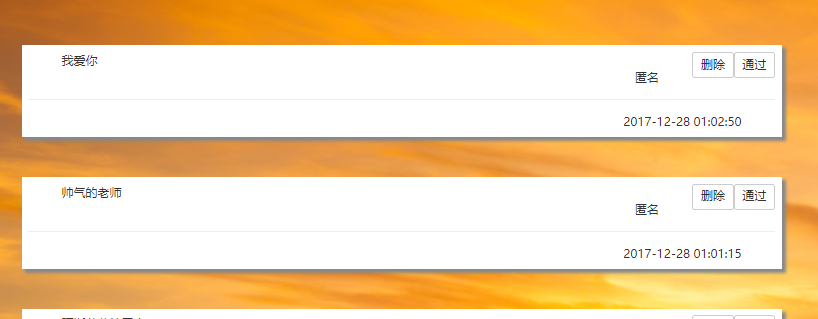


图4-1

将“我爱你”进行删除操作，结果如下图

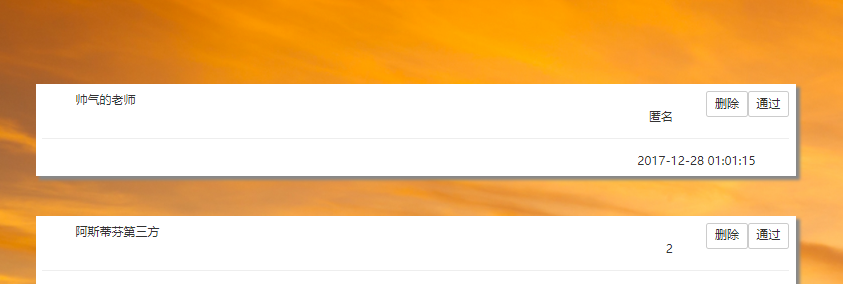


图4-2

由上图可见该条留言审核通过。再对“帅气的老师”进行通过操作，并退回到主界面中，结果如下图。

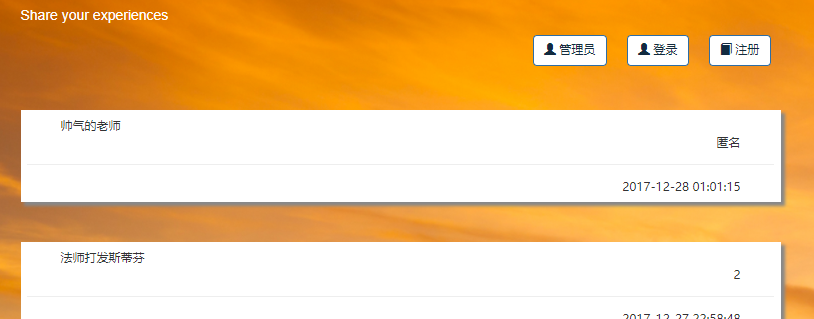


图4-3

由上图可见说明消息发送、审核、删除功能测试成功。其它功能本人也进行测试，结果都符合要求，不再一一举例。

5总结

本次课程设计，使我受益匪浅，我学到很多以前不会和没有接触过得东西，加深对RESTful API 的理解，通过对web网站开发进行前后端分离的实践，锻炼良好的软件设计思维。掌握web应用发布的基本原理，熟悉常用web服务器的配置和使用。以前重来没有过软件前后端分离的概念，同时使我加深了对软件开发的的认识，同时了解到一个强大的软件是如何被设计出来的，不同的设计人员可以通过不同的接口去进行不同的操作。前后端的完全分离的好处是开发人员可以相对独立的开发软件，后端服务器配置的接口，任何应用程序能够在使用这些接口的基础上实现各个功能。

在本次课程设计过程中，我也遇到许多的问题，第一次接触这一陌生的概念，使我不知道该冲何处下手。由于知识和经验的欠缺，软件的开发很吃力，开发出来的软件的功能还不算太完善。在设计的过程中也遇到了很多困难，由于对整个开发方式的概念模糊，给后来系统的具体实现带来了很大的困难，中间做了许多无用功。

在设计过程中查找了许多的书籍和有关资料，并且在网上查询许多的实例，与同学相互交流。从这些方法中我获取到很多的知识，使我的编码能力提高许多，同时学习到对前后端分离，将后端服务器的处理方法封装成一个个的接口的方法。这些知识以前没有接触到，学习这种软件编程的方法还是很有必要的。在这次设计的过程中发现了自己很多不足之处，意识到自己对之前所学过的知识理解得不够深，掌握不够牢，有待加强。既巩固了旧知识，又学习到了新知识，我在本次课程设计中收获颇多。

附录

@Path("/user")

**public** **class** dengluoperator {

@Path("/denglu")

@GET

@Produces(MediaType.***APPLICATION\_JSON***) //返回数据的格式

**public** denglutable denglu(

@QueryParam("name") String name,

@QueryParam("password") String password)

{

denglutable user=**new** denglutable();

user.setName(name);

user.setPassword(password);

System.***out***.println(user.toString());

Session session = HibernateUtils.*getCurrentSession*();

Transaction ts = session.beginTransaction();

Criteria c = session.createCriteria(denglutable.**class**);

c.add(Restrictions.*eq*("name", user.getName()));

denglutable user1 = (denglutable) c.uniqueResult();

**if**(user1 != **null**){

**if**(user1.getPassword().equals(user.getPassword())){

**return** user1;

}

}

**return** **null**;

}

//管理员登录

@Path("/guanlidenglu")

@GET

@Produces(MediaType.***APPLICATION\_JSON***) //返回数据的格式

**public** denglutable guanlidenglu(

@QueryParam("name") String name,

@QueryParam("password") String password)

{

denglutable user=**new** denglutable();

user.setName(name);

user.setPassword(password);

System.***out***.println(user.toString());

Session session = HibernateUtils.*getCurrentSession*();

Transaction ts = session.beginTransaction();

Criteria c = session.createCriteria(denglutable.**class**);

c.add(Restrictions.*eq*("name", user.getName()));

c.add(Restrictions.*eq*("status", 1));

denglutable user1 = (denglutable) c.uniqueResult();

System.***out***.println(user1.toString());

**if**(user1 != **null**){

**if**(user1.getPassword().equals(user.getPassword())){

**return** user1;

}

}

**return** **null**;

}

//注册

@Path("/zhuche")

@GET

@Produces(MediaType.***APPLICATION\_JSON***) //返回数据的格式

**public** denglutable zhuche(

@QueryParam("name") String name,

@QueryParam("password") String password)

{

denglutable user=**new** denglutable();

user.setName(name);

user.setPassword(password);

user.setStatus(0);

Session session = HibernateUtils.*getCurrentSession*();//操作数据库

Transaction ts = session.beginTransaction();

Criteria c = session.createCriteria(denglutable.**class**);

c.add(Restrictions.*eq*("name", user.getName()));

denglutable user1= (denglutable) c.uniqueResult();

**if**(user1 != **null**){

**return** **null**;

}

**else**{

session.save(user);

ts.commit();

**return** user;

}

}

@Path("/message")

public class message {

//留言

@Path("/sendmessage")

@GET

@Produces(MediaType.APPLICATION\_JSON) //返回数据的格式

public List<messagetable> send(

@QueryParam("message") String mess,

@QueryParam("sendMessageUserName") String name)

{

Date day=new Date();

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

messagetable message=new messagetable();

if(name.equals(""))

{

message.setName("匿名");

}

else

{

message.setName(name);

}

message.setMessage(mess);

message.setTime(df.format(day));

//System.out.println(message.toString());

Session session = HibernateUtils.getCurrentSession(); //@连接数据库

Transaction ts = session.beginTransaction(); //开启事务

session.save(message);

Criteria c = session.createCriteria(messagetable.class); //创建查询

c.add(Restrictions.eq("respond", 1));

c.addOrder(Order.desc("time")); //时间倒序

List<messagetable> list = c.list();

ts.commit();// 提交事务

return list;

}

//获取审核留言通过接口

@Path("/getmessage")

@GET

@Produces(MediaType.APPLICATION\_JSON) //返回数据的格式

public List<messagetable> get()

{

Session session = HibernateUtils.getCurrentSession();

Transaction ts = session.beginTransaction();

Criteria c = session.createCriteria(messagetable.class);

c.add(Restrictions.eq("respond", 1));

c.addOrder(Order.desc("time"));

List<messagetable> list = c.list();

ts.commit();

return list;

}

//获取未审核的留言

@Path("/getnorespondmessage")

@GET

@Produces(MediaType.APPLICATION\_JSON) //返回数据的格式

public List<messagetable> getnorespond()

{

Session session = HibernateUtils.getCurrentSession();

Transaction ts = session.beginTransaction();

Criteria c = session.createCriteria(messagetable.class);

c.add(Restrictions.eq("respond", 0));

c.addOrder(Order.desc("time"));

List<messagetable> list = c.list();

ts.commit();

return list;

}

//审核

@Path("/respond")

@GET

@Produces(MediaType.APPLICATION\_JSON) //返回数据的格式

public List<messagetable> respond (@QueryParam(value="a") int id)

{

Session session = HibernateUtils.getCurrentSession();

Transaction ts = session.beginTransaction();

messagetable message = session.load(messagetable.class, id);

message.setRespond(1);

session.update(message);

Criteria c = session.createCriteria(messagetable.class);

c.add(Restrictions.eq("respond", 0));

c.addOrder(Order.desc("time"));

List<messagetable> list = c.list();

ts.commit();

return list;

}

//删除留言

@Path("/delete")

@GET

@Produces(MediaType.APPLICATION\_JSON) //返回数据的格式

public List<messagetable> delete (@QueryParam(value="b") int id)

{

Session session = HibernateUtils.getCurrentSession();

Transaction ts = session.beginTransaction();

messagetable message = session.load(messagetable.class, id);

session.delete(message);

Criteria c = session.createCriteria(messagetable.class);

c.add(Restrictions.eq("respond", 0));

c.addOrder(Order.desc("time"));

List<messagetable> list = c.list();

ts.commit();

return list;

}

}