****

软件开发实训

课程设计报告

题 目： **留 言 板**

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： 蓝琳琪

学 号： 1400310117

指导教师： 梁 海

2017 年 12月 24 日

**1、课设任务**

1. 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）。
2. 留言只有在管理员回复以后才能显示在页面上，界面设计美观大方。

**2、详细设计**

**2.1 数据库设计**

**2.1.1 数据表设计**

表2.1.1 用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名** | **类型** | **宽度** | **是否主码** | **是否为空** | **描述** |
| **idclient** | int |  | PK | 否 | 用户ID |
| name | VChar | 100 |  | 否 | 用户名 |
| password | VChar | 100 |  | 否 | 密码 |

表2.1.2 管理员表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名** | **类型** | **宽度** | **是否主码** | **是否为空** | **描述** |
| **idmanage** | int |  | PK | 否 | 管路员ID |
| name | VChar | 100 |  | 否 | 管理员名 |
| password | VChar | 100 |  | 否 | 密码 |

表2.1.3 留言信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名** | **类型** | **宽度** | **是否主码** | **是否为空** | **描述** |
| **idmessage** | int |  | PK | 否 | 信息ID |
| idclient | int |  |  | 否 | 用户ID |
| idmanage | int |  |  | 否 | 管理员ID |
| mes | Vchar | 200 |  | 否 | 信息体 |
| flag | int |  |  | 否 | 审核标记 |

**2.1.2 E-R图设计**

用户

管理员

信息核查

信 息

信息发布

留 言

图2.1.1 总体E-R图

**2.2 结构设计**

留 言 板

注册界面

用户注册

管理员注册

登录界面

用户登录

管理员登录

信息评价

信息审核

图2.2 层次方框图

**3、系统设计**

**3.1 Dao层设计**

（1）Dao层实现与数据库的直接相连，实现对数据库数据的直接操作，本项目所需要对用户表、管理员表、信息表的处理，在包含数据库的工程项目开发，DAO设计的好坏，直接影响到整个工程的结构，一个好的Dao设计，可以大大减少工程的代码量。

DAO <T> 类

（实现增、删、改、查方法）

xxxDao 接口

定义xxx所需方法

xxxDaoImpl 实现类

实现接口

继承

图4.1 DAO设计图

DAO<T> 类是利用泛型实现通用于各个实现类的增、删、改、查方法；xxxDao 接口定义所需方法，这些方法主要是对特定表的相关处理；xxxDaoImpl类调用DAO<xxx>中通用方法实现xxxDao接口的方法。Servlet层只需调用xxxDaoImpl实现类即可，若需要修改，先修改xxxDao接口，系统会报xxxDaoImpl类的错误，再实现所需方法即可。

1. **代码实现**：
2. DAO<T>类的主要通过泛型、反射与一些组件的应用，实现增、删、改、查的通用；

public class AllDao<T> {

private Class<T> clazz;

private QueryRunner queryRunner=new QueryRunner();//组件使用

@SuppressWarnings("unchecked")

public AllDao() { //泛型实现各类映射

Type type=getClass().getGenericSuperclass();

if(type instanceof ParameterizedType){

ParameterizedType pt=(ParameterizedType)type;

Type[] t=pt.getActualTypeArguments();

if(t[0] instanceof Class){

clazz=(Class<T>)t[0];

}

}

}

public List<T> getForList(String sql,Object...objects) throws SQLException{ //查询返回的数据集

Connection con=null;

try{

con=DaoToot.getConnect();

return queryRunner.query(con,sql,new BeanListHandler<>(clazz),objects);

}catch(SQLException e){

throw new SQLException();

}finally{

DaoToot.release(con);

}

}

@SuppressWarnings("unchecked")

public <E> E getForValue(String sql,Object...objects) throws SQLException{ //查询返回的单个值

Connection con=null;

try{

con=DaoToot.getConnect();

return (E)queryRunner.query(con, sql,new ScalarHandler(),objects);

}catch(SQLException e){

throw new SQLException();

}finally{

DaoToot.release(con);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\* 查询返回的单个类实例 \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public T get(String sql,Object...objects) throws SQLException{

Connection con=null;

try{

con=DaoToot.getConnect();

return queryRunner.query(con, sql, new BeanHandler<>(clazz),objects);

}catch(SQLException e){

throw new SQLException();

}finally{

DaoToot.release(con);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 增、删、改 操作通用 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void update(String sql,Object...objects) throws SQLException {

Connection con=null;

try{

con=DaoToot.getConnect();

// System.out.println(con);

queryRunner.update(con, sql, objects);//@@@@@@@@@@@@@@@

// System.out.println("2");

}catch(SQLException e){

throw new SQLException();

}finally{

}

}

}

1. 实现对应DAO<T>的相应接口 UserDao；

public interface UserDao {

//发布信息存入数据库

public void save(User user) throws SQLException;

//获取用户匹配个数

public long getCount(Client c) throws SQLException;

//客户注册存入数据库

public void saveClient(Client client) throws SQLException;

public long getCount1(Client c) throws SQLException;

}

1. 实现对应UserDA接口相应方法；

@Override

public void save(User user) throws SQLException{

// TODO Auto-generated method stub

String sql="insert into user.message(mes) values(?)";

update(sql,user.getMes());

}

@Override

public long getCount(Client c) throws SQLException {

// TODO Auto-generated method stub

String sql="select count(\*) from user.login where name=? and password=?";

return getForValue(sql, c.getName(),c.getPassword());

}

@Override

public long getCount1(Client c) throws SQLException {

// TODO Auto-generated method stub

String sql="select count(\*) from user.login where name=?";

return getForValue(sql, c.getName());

}

@Override

public void saveClient(Client client) throws SQLException {

// TODO Auto-generated method stub

if(getCount1(client)!=0){

return;

}

String sql="insert into user.login(name,password) values(?,?)";

update(sql,client.getName(),client.getPassword());

}

**3.2 中间层设计**

在中间层的设计，主要实现与Dao层的结合，实现与View层的交互，在中间层，我们主要采用RESTful实现前端与后端的分离，基于这个风格设计的软件可以更简洁，更有层次，更易于实现缓存等机制。

（1）首先要先了解RESTful API的机制，通过获取的POST或GET请求，来进行相应的处理的操作。

（2）通过URL来传输相应的数据，可调用Dao层的方法进行操作，也可以直接处理后返回View层XML数据或json数据。

（3）RESTful代码实现：

// 这里@Path定义了类的层次路径。

// 指定了资源类提供服务的URI路径。

@Path("UserInfoService")

**public** **class** UserInfo {

// @GET表示方法会处理HTTP GET请求

@GET

// 这里@Path定义了类的层次路径。指定了资源类提供服务的URI路径。

@Path("/name/{i}")

// @Produces定义了资源类方法会生成的媒体类型。

@Produces(MediaType.*TEXT\_XML*)

// @PathParam向@Path定义的表达式注入URI参数值。

**public** **void** userName(@PathParam("i") String i) {

String name = i;

User user=**new** User();

user.setMes(name);

UserDao udi=**new** UserDaoImpl();

**try** {

udi.save(user);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

//return "<User>" + "<Name>" + name + "</Name>" + "</User>";

//return name;

}

//@GET表示方法会处理HTTP GET请求

@GET

//这里@Path定义了类的层次路径。指定了资源类提供服务的URI路径。

@Path("/name1/{text}")

//@Produces定义了资源类方法会生成的媒体类型。

@Produces(MediaType.*TEXT\_XML*)

//@PathParam向@Path定义的表达式注入URI参数值。

**public** **void** login(@PathParam("text") String text) {

String n = text;

User user=**new** User();

user.setMes(n);

UserDao udi=**new** UserDaoImpl();

**try** {

udi.save(user);

} **catch** (SQLException e) {

e.printStackTrace();

}

//return;

}

@GET

@Path("/age/{j}&{k}")

@Produces(MediaType.*TEXT\_XML*)

**public** **void** userAge(@PathParam("j") String j , @PathParam("k") String k) {

String name=j;

String password=k;

Client client=**new** Client();

client.setName(name);

client.setPassword(password);

UserDao udi=**new** UserDaoImpl();

**try** {

udi.saveClient(client);

} **catch** (SQLException e) {

e.printStackTrace();

}

//return "<User>" + "<Name>" + name + "</Name>" +"<password>" + password + "</password>" + "</User>";

}

}

**3.3 View层设计**

视图的设计，主要决定该系统给用户的感受。本系统采用HTML实现的界面，主要通过JavaScript与css实现界面的动态与样式。通过POST与GET发送往中间层设计。在View层有登录界面与信息发布界面。

（1）登录界面：主要有用户登录与用户注册，登录需要同数据库联系，查询数据库是否存在该用户且密码是否匹配。注册也需要同数据库联系，查询用户是否同名，以判断是否可注册；

（2）信息发布界面：用于用户信息发布，采用Jqurey对界面的动态修改，并在click事件中采用location.href=URL跳转到URL,即调用中间层RESTful进行操作。

（3）代码实现：

1、登录界面相关

<div id=*"login"*>

<form action=*"UserServlet"* method=*"post"* >

<fieldset>

<legend><span class=*"logintext"*>登 录 入 口</span></legend>

<span class=*"text"* >账号：</span><input type=*"text"* value=*"lan"* name=*"name"* id=*"first"*><br />

<span class=*"text"* >密码：</span><input type=*"password"* value=*"123"* name=*"password"* id=*"two"*><br />

<input type=*"submit"* value=*"登 录"* id=*"loginsubmit"*/>&nbsp;&nbsp;<input onclick="out1()" value=*"注 册"* type=*"button"* id=*"res"*>

</fieldset>

</form>

</div>

<script type=*"text/javascript"*>

**function** out1(){

**var** oneText=document.getElementById("first").value;

**var** twoText=document.getElementById("two").value;

// alert(oneText);

location.href="http://localhost:8989/RESTfulWS1/rest/UserInfoService/age/"+oneText+"&"+twoText;

}

</script>

1. 信息发布界面相关

<script type=*"text/javascript"* src=*"use/main.js"*></script>

<script type=*"text/javascript"* src=*"use/sinaFaceAndEffec.js"*></script>

<script type=*"text/javascript"*>

// 绑定表情

$('.face-icon').SinaEmotion($('.text'));

// 测试本地解析

**function** out() {

**var** inputText = $('.text').val();

$('#info-show ul').append(reply(AnalyticEmotion(inputText)));

//location.href="http://localhost:8989/RESTfulWS/rest/UserInfoService/aaa";

location.href="http://localhost:8989/RESTfulWS1/rest/UserInfoService/name1/"+inputText;

}

**var** html;

**function** reply(content){

html = '<li>';

html += '<div class="head-face">';

<!--html += '<img src="images/1.jpg" / >';-->

html += '<img src="use/1.jpg" / >';

html += '</div>';

html += '<div class="reply-cont">';

html += '<p class="username">小小红色飞机</p>';

html += '<p class="comment-body">'+content+'</p>';

html += '<p class="comment-footer">2016年10月5日　回复　点赞54　转发12</p>';

html += '</div>';

html += '</li>';

**return** html;

}

</script>

**4、系统实现**

**4.1 功能实现**

（1）View层登录界面实现



图4.1.1 登录注册图

（2）View层信息发布界面实现

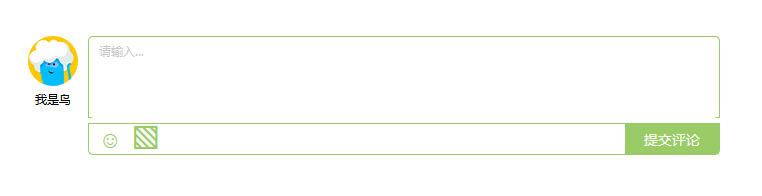


图4.1.2 信息发布1



图4.1.3 信息发布2

**4.2 调试分析与处理**

在编程的过程出现过，我往往通过很长时间才能将其处理，这也正说明了我在这方面知识的缺失，以下是我在编程中出现的问题：

1. 在编写View层代码，主要js与css,我曾尝试用Jqurey获取input文本框的value值，发现什么弄都获取不了，也上网查了相关资料，问题依旧未能解决，最终只能依靠JavaScript获取value值，依我判断，Jqurey本身存在缺陷，Jqurey虽然简洁，但依旧没有JavaScript强大，无法获取表单中的数据，用表单外按钮的点击事件。
2. [在编写中间代码时，主要GET和POST两种形式URL，通过对URL解析，提取“/”后的数据，在RESTful进行处理，开始我以为这只是普通的java程序，便设置返回为任意类型，结果不然，应是与@Produces(MediaType.TEXT\_XML)相关，且返回的是xml数据json数据，或者无返回值。](mailto:在编写中间代码时，主要GET和POST两种形式URL，通过对URL解析，提取\“/\”后的数据，在RESTful进行处理，开始我以为这只是普通的java程序，便设置返回为任意类型，结果不然，应是与@Produces(MediaType.TEXT_XML)相关，且返回的是xml数据json数据，或者无返回值。)
3. 在Dao层设计时，没能对反射机制、泛型的准确使用，导致了很多错误，还有在对数据库的相关操作，需用到相应的SQL语句，对于不同数据库系统,SQL语句可能有所不同，我也这一方面吃了很多亏。