Lab 8 设计实现留言板

实验目的

1) 复习 MVC 设计模式
2) 复习 serlvet/jsp
3) 熟练掌握 JDBC 连接 MySQL
4) 熟练掌握 MySQL 的操作
5) 熟练掌握处理各种关于中文的乱码显示问题

实验任务

通过设计 JSP、Servlet、JavaBean 使用 MVC 模式实现留言的添加、删除和修改，页面中留言的添加、删除和修改通过 servlet 实现，页面显示留言内容通过 JSP 访问 JavaBean 实现，在 JSP 页面中使用 JSTL，不能出现 Java 代码片段。留言包括用户名、标题、留言日期、留言时间以及内容，其中相关信息可以使用中文；添加和修改留言时只显示用户名、标题、留言内容，留言日期和留言时间在 servlet 中由程序生成。留言内容通过 JavaBean 存入 MySql 中建立的表中。本实验中 MVC 的组成如下：

M package：bean
message.java 定义消息的内容以及操作
messageDB.java 定义与消息有关的数据库操作
V index.jsp 主页面，显示当前所有留言
Info.jsp 添加和修改消息的页面
C package: filter
CharacterEncodingFilter.java 过滤器
Package: servlet
MsgProcess.java 处理留言的添加、修改和删除

实验环境

数据库 MySQL 5
开发工具 不限（推荐 MyEclipse）
Servlet/JSP 容器：Apache Tomcat 5.5 或以上
JDK：Sun JDK 1.5 或以上
浏览器：Internet Explorer 6 或以上，Firefox 1.5 或以上

实验提交物

本次试验需要在 5：00 之前让 TA 检查，记录，上传所有源码文件以及 WAR 文件

实验步骤

1．中文乱码问题

乱码问题是由于未对中文进行特殊的编码处理，由于编码可采用不同的编码方式，包括 UTF-8、GBK、GB2312 都是支持中文的，本实验中统一采用 UTF-8。

1） JSP 页面中文乱码解决方法

在各个 JSP 页面中加入如下

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
```

2） 表单提交中乱码解决方法

使用 filter 解决，过滤器的基本原理就是对于每一个用户请求，都必须经过过滤器的处理才能继续发生到目的页面中。在 JSP 中，以 POST 方式提交的表单本质上就是封装 request 对象中的，而 request 对象是必须要经过过滤器的处理的，所以对于中文乱码问题，可以在 filter 中对所有的请求进行编码格式的处理。

文件 `CharacterEncodingFilter.java` 为中文处理过滤器，通过继承 Filter 类实现。过滤器实现处理中文的方式为：

a) 在初始化函数 init 中加载配置文件中定义的编码类型，具体方法为：

```
FilterConfig filterConfig;//定义成员变量
String encodingName; //定义成员变量
boolean enable; //定义成员变量
this.encodingName = this.filterConfig.getInitParameter("encoding");
String strIgnoreFlag = this.filterConfig.getInitParameter("enable");
        if(strIgnoreFlag.equalsIgnoreCase("true")){
            this.enable = true;
        }
        else{
            this.enable = false;
        }
```

b) 在 `doFilter` 函数中实现过滤中文

```
if(this.enable){
            request.setCharacterEncoding(this.encodingName);
        }
        chain.doFilter(request, response);
```

c) 修改配置文件，在 web-inf/web.xml 文件中添加如下配置信息

```
<filter>
    <filter-name>CharacterEncodingFilter</filter-name>
    <filter-class>filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
        <param-name>enable</param-name>
        <param-value>true</param-value>
    </init-param>
  </filter>
```

```
  <filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
     </filter-mapping>
```

3) 数据库乱码解决方法

在 MySql 中建立数据库和表示统一使用 UTF-8 编码。

数据库建立连接时使用 UTF-8 编码：

```
"jdbc:mysql://xx.xx.xx.xx:3306/lab9?user=xx&password=xx&useUnicode
=true&characterEncoding=utf-8";
```

注：xx.xx.xx.xx 为所访问 MySql 的 IP

2. 建立数据库

在 MySQL 中建立数据库 lab9，使用 uft8 编码；

```
CREATE DATABASE `lab9`
CHARACTER SET 'utf8'
COLLATE 'utf8_general_ci';
```

在此数据库下建立表message，使用uft8编码；

```
CREATE TABLE `message` (
  `Id` int(11) NOT NULL auto_increment,
  `userName` varchar(20) default NULL,
  `date` varchar(20) default NULL,
  `time` varchar(20) default NULL,
  `title` varchar(20) default NULL,
  `content` varchar(255) default NULL,
  PRIMARY KEY  (`Id`)
) ENGINE=MyISAM AUTO_INCREMENT=13 DEFAULT CHARSET=utf8;
```

3. 数据库的连接与操作

首先定义留言的实体Bean，创建包bean，并在此包中新建文件message.java，此实体bean对于操作数据库message表的六个字段

Id, userName, date, time, title, content

提供相应的set和get方法，由于id字段在表中定义的是自增的，故在此实体中不提供id的set方法。

数据库的操作在独立的Bean中定义，数据库连接通过JDBC连接，需要第三方库的支持，此库有TA在ftp上提供，下载后，将mysql-connector-java-5.0.5-bin.jar保存到\WEB-INF\lib目录下

在bean包中新建文件messageDB.java，用于操作数据库，实现留言的增加、删除、修改以及查询。以下给出此文件的代码，阴影部分需要同学们根据提示补全。

```
package bean;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

public class messageDB{
    private message msg;
    private List<message> msgs;
    Connection conn;
    Statement stat;
    ResultSet rs;
    public messageDB(){
        msgs = new ArrayList<message>();
        msg = new message();
    }

    void openDB(){
        String url = 定义连接方式
        try {
                Class.forName("com.mysql.jdbc.Driver");
```

```
} catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
            try {
                this.conn = DriverManager.getConnection(url);
                stat = conn.createStatement();
            } catch (SQLException e) {
                e.printStackTrace();
            }
    }

    void closeDB(){
        try {
            conn.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    //此函数返回当前message表中所有留言记录到一个链表中，用于显示
    public List getMsgs(){
          打开数据库

        String sql =    数据库查询语句
        try {
            rs = stat.executeQuery(sql);

            while(rs.next()){
                String str = rs.getString(6);
                str = str.replace("\r\n", "<br>");
                //System.out.print(str);
                msg = new message(rs.getInt(1),rs.getString(2),
rs.getString(3), rs.getString(4), rs.getString(5),str);
                msgs.add(msg);
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }finally{
            try {
                if(rs!=null)
                rs.close();
                if(stat !=null)
                stat.close();
```

```
            e.printStackTrace();
            }
            closeDB();
        }
        return msgs;
    }
   //此函数根据给定ID修改数据库中的一条留言记录
    public void setMsg(message innerMsg){
        openDB();
        int id = innerMsg.getId();
        String content = innerMsg.getContent();
        String date = innerMsg.getDate();
        String time = innerMsg.getTime();

        String sql =    给出 SQL 更新语句
        try {
            stat.execute(sql);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }finally{
            try {
                stat.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            closeDB();
        }
    }
//向数据库中添加一条留言记录
public void addMsg(message innerMsg){
        openDB();
        String sql =    给出 SQL 更新语句
```

```
    try {
            stat.execute(sql);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }finally{
            try {
                stat.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            closeDB();
        }
    }
    //根据给定ID删除数据库中的一条留言记录
    public void delMsg(int id){
        openDB();

        String sql =
        try {
            stat.execute(sql);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }finally{
            try {
                stat.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            closeDB();
        }
    }
public void setMsgId(String id){
        int msgId = Integer.valueOf(id);
        this.msg.setId(msgId);
    }

    public message getMsg(){
        return this.getMsg(this.msg.getId());
    }
```

根据 ID 删除一条记录语句

```
public message getMsg(int id){
        openDB();

        String sql = 根据 ID 查询一条记录

        try {
            rs = stat.executeQuery(sql);
            while(rs.next()){
                msg = new message(rs.getInt(1),
rs.getString(2),rs.getString(3),rs.getString(4),rs.getString(5),rs
.getString(6));
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }finally{
            try {
                rs.close();
                stat.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            closeDB();
        }
        return msg;
    }
}
```

4. 处理留言的添加、删除和修改

在自定义包servlet中建立文件名为MsgProcess.java的servlet，其dopost方法定义如下补全阴影部分

```
        String action = request.getParameter("action");
        String innerID = request.getParameter("id");
        int id = -1;
        if(innerID != null && innerID.length() != 0)
            id = Integer.valueOf(innerID);
        String name = request.getParameter("name");

        Date da = new Date();
        String[] tmp = da.toLocaleString().split(" ");
        String date = tmp[0];
        String time = tmp[1];
```

```
String title = request.getParameter("title");
        String content = request.getParameter("content");
        //content = content.replaceAll("\\r\\n", "<br>");
        bean.message msg;
        bean.messageDB msgDB = new bean.messageDB();
        if(action != null){
            if(action.equals("add")){
                通过 messageDB 向数据库中添加一条记录
            }
            else if(action.equals("del")){
                通过 messageDB 以及 ID 从数据库中删除一条记录
            }
            else if(action.equals("mod")){
                通过 messageDB 以及 ID 从数据库中查询一条记录
                根据获取的参数修改此记录，并更新数据库
            }
        }
        response.sendRedirect("../index.jsp");
```

5，留言在前端的显示、添加、修改以及删除

页面的显示通过JSP直接调用JavaBean实现，以及通过JSTL操作Bean的内容，添加支持JSTL的库可参考前次实验说明，通过JSTL显示JavaBean中的内容的方法

```
<jsp:useBean id="msgDB"  class="bean.messageDB" />
<jsp:setProperty name="msgDB" property="*"/>
      <c:set var="msgs" value="${msgDB.msgs}"/>
<c:forEach items="${msgs}" var="msg">
```

显示留言页面index.jsp代码如下，补全阴影部分，不可以使用Java代码片段

```
    设置编码方式，调用 Bean：messageDB，  初始化此 Bean，  定义 JSTL 变量
<html>
  <head>
    <title>Message Board</title>
    <script language="javascript" src="assist.js">
    </script>
  </head>
   <body>
```

```
    <form >
    <table width="674" height="85" border="1">
    <tr>
    <td>用户名</td>
    <td>日期</td>
    <td>时间</td>
    <td>标题</td>
    </tr>
    JSTL 循环开始
<tr name="innertr" id="${ 留言 ID }"
onMouseOver="showContent('${ 留言内容 }')"
onMouseOut="showContent(null)"
onClick="showButton(${ 留言 ID })"><td>

    通过 JSTL，循环打印数据库的所有记录

   <tr>
    <td colspan="1"><input type="button" name="add" value="添加" onClick="addMsg()">
    </td><td colspan="1"><div id="divId1"></div></td>
    <td colspan="1"><div id="divId2"></div></td>
    <td> </td>
    </tr>
    </table>
    <div id="divId3"></div>
    </form>
   </body>
</html>
```

以上文件中使用的 JS 文件有 TA 提供，页面的显示效果如下

用户名	日期	时间	标题
大侠	2008-6-5	22:43:26	问候
虾米	2008-6-5	22:44:57	灌水中
添加			

鼠标点击留言行时，该行会变色，并且显示删除和修改按钮，鼠标悬停在某行时，在下方显示该留言的内容，留言内容可以多行显示，这些功能通过 JS 实现，JS 文件 TA 提供，需要注意此文件中含有中文，因而此文件的编码方式需要改为 UTF-8。

修改和添加留言的页面通过 info.jsp 文件实现，文件通过 JSTL 判断当前的 action，若为修改，用户名和标题不可更改，其余都可修改，所有内容中英文兼可，留言内容可以输入为多行。页面不可以出现 Java 代码片段。页面效果图如下

提示：

JSTL 获取 request 的方法

```
<c:set var="action" value="${requestScope.action}" />
<c:set var="id" value="${requestScope.id}" />
```

页面操作 Bean 的方法

```
<jsp:useBean id="msgDB"  class="bean.messageDB" />
<jsp:setProperty name="msgDB" property="msgId" value="${param.id}"/>
```

判断当前操作的方法

```
<c:if test="${param.action=='add'}" >
```

此页面的表单通过 post 方法提交到 servlet/MsgProcess 处理

由于本页面需要向 servlet 传递当前的留言的 ID 和当前的 action（添加/删除），因而需要定义两个隐藏字段，方法如下

```
<input type="hidden" name="id" value="${param.id}">
    <input type="hidden" name="action" value="${param.action}">
```

付 JS 文件　assist.js

```
var GlobalID=-1;
    function showButton(id){
        var element = document.getElementById("divId1");
        element.innerHTML = "<input name=\"del\" type=\"button\"
onClick=\"mod(" + id + ")\" value=\"修改\">";
        element = document.getElementById("divId2");
        element.innerHTML = "<input name=\"del\" type=\"button\"
onClick=\"delMsg(" + id + ")\" value=\"删除\">";
        if(GlobalID!=-1){
            element = document.getElementById(GlobalID);
            element.style.background="#FFFFFF";
        }
```

```
        element = document.getElementById(id);
        element.style.background = "#999999";
        GlobalID = id;
    }

    function showContent(content){
        var element = document.getElementById("divId3");
        if(content != null)
            element.innerHTML = "留言内容：<br>" + content;
        else
            element.innerHTML = "";
    }

    function addMsg(){
        window.location.href("info.jsp?action=add");
    }

    function mod(id,name,title,content){
        document.location.replace("info.jsp?action=mod&id=" + id);
    }

    function delMsg(id){
        document.location.replace("servlet/MsgProcess?action=del&id="
+ id);
        }
```