第 8 章　发表回复功能的实现

本章知识要点：

- 在线编辑器组件 FCKeditor 的使用；
- 数据库处理通用类的使用。

8.1　任务提出

发帖包括“发表新帖”和“发表回复”两种情况，即发表新的话题帖子，或针对已有的帖子参与发表自己的回复意见。

本章仅讲解回帖功能的实现过程。“发表新帖”功能的实现过程与此相似，源码可参考完整的论坛演示项目 bbs 的 WebRoot 目录中的 newPost.jsp 和 savePost.jsp。

8.2　制作回复帖内容输入页面

发表回复或新帖时，帖子内容的字数可能较多，一般可通过 HTML 的多行文本框标签＜textarea＞来提供输入，若需要提供较为丰富的编辑功能，则可选择在线编辑器来实现。下面将学习多行文本框与在线编辑器的使用。

8.2.1　HTML 标签——textarea

HTML 中的 textarea 标签可提供页面上的一个多行文本框，其格式为：

```
＜textarea name="" cols="" rows=""＞初始内容＜/textarea＞
```

name 表示该多行文本框的名称，而 cols 和 rows 属性分别表示显示区域的列数和行数，若实际输入的内容超出了这个限制，则通常会出现水平或垂直方向的滚动条。在标签＜textarea＞和＜/textarea＞之间可以填写文字，它们将作为文本框的初始内容显示在文本框中。

例 8.1　多行文本框的使用。

主要代码　WebRoot\8\8_1\textarea.html

```
1:  <body>
2:    <form  action="save.jsp">
3:      <textarea name="content" cols="60" rows="10">在此填写意见
4:    </textarea>
5:    <br/>
6:    <input type="submit"   value="确定"/>
7:    </form>
8:  </body>
```

该例的效果如图 8-1 所示。

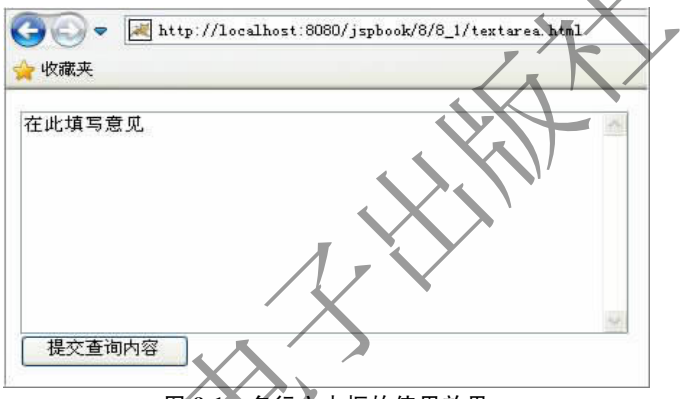

图 8-1　多行文本框的使用效果

8.2.2　在线编辑器 FCKeditor 的使用

我们通过 HTML 的 textarea 标签，可以输入较长的文本内容，但是所输入的内容是纯文本的格式，比较单一，若需要控制输入内容的字体、字号、颜色，可以通过编写 JavaScript 程序实现这样的在线编辑功能。不过，由于这样的需求比较普遍，因此目前已有多款的在线编辑器软件可供使用，FCKeditor 是其中非常著名的一款。

FCKeditor 是一款功能强大的免费软件，可以通过网站下载其最新版本，下载地址为 http://www.fckeditor.net/download 或 http://ckeditor.com/download。这里以 2.6 版为例说明其使用方法。

从官方网上下载 FCKeditor 压缩包，将得到的文件 FCKeditor_2.6.3.zip 解压，得到 fckeditor 目录，按照以下步骤在项目里添加 FCKeditor，并在页面中使用：

(1)将 fckeditor 目录的内容只保留如图 8-2 所示的 4 个文件和 1 个文件夹，然后把 fckeditor 目录复制到 WebRoot 文件夹下。

图 8-2 FCKeditor 的主要文件

其中的 fckeditor.js 是其核心文件，fckconfig.js 则是配置文件。

(2)在需要使用在线编辑器的页面里按以下步骤添加代码：

①在标签对<head></head>间添加对 js 文件的引用：

```
<script type="text/JavaScript" src="<%=request.getContextPath()%>/fckeditor/fckeditor.js"></script>
```

②在标签对<body></body>间需要显示在线编辑器的地方添加：

```
<script type="text/JavaScript">
        var sBasePath="<%=request.getContextPath()%>/fckeditor/";
        var oFCKeditor = new FCKeditor('pcontent') ;
        oFCKeditor.BasePath = sBasePath ;
        oFCKeditor.Height = 400 ;
        oFCKeditor.Value = '' ;
        oFCKeditor.Create() ;
</script>
```

在页面中引入对 fckeditor.js 文件的引用时，需指明被引用文件的路径。

代码 var oFCKeditor = new FCKeditor('pcontent')表示输入框的名称是 pcontent，当数据提交到指定页后，可在接收页面中使用 request.getParameter("pcontent")获取它的值。

FCKeditor 在线编辑器的界面如图 8-3 所示：

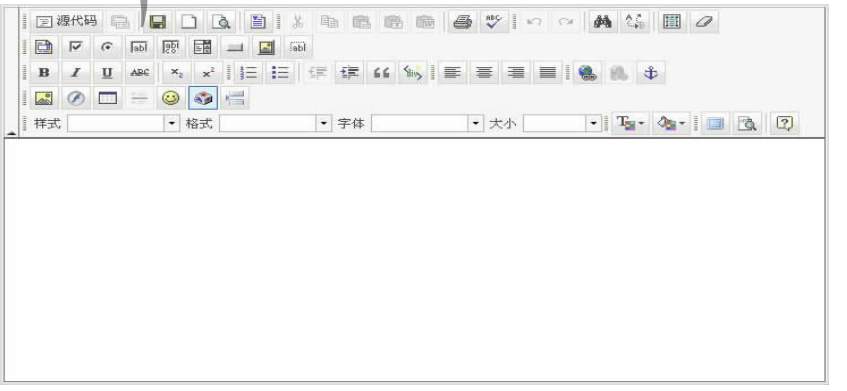

图 8-3 FCKeditor 的默认运行效果

可以看到它的编辑功能非常强大，几乎可以与 MS Office 套件中的 Word 媲美。不过，实际使用时往往不需要这么复杂的功能。通过修改 fckeditor 目录下的配置文件 fckconfig.

js,可以定制自己的工具条。

打开此文件,找到其中的 FCKConfig. ToolbarSets 配置段,这部分代码形如:

```
1:  FCKConfig.ToolbarSets["Default"] = [
2:  ['Source','DocProps','-','Save','NewPage','Preview','-','Templates'],
3:  ...
4:  ];
5:
6:  FCKConfig.ToolbarSets["Basic"] = [
7:      ['Bold','Italic','-','OrderedList','UnorderedList','-','Link',
    'Unlink','-','About']
8:  ];
9:
10:  FCKConfig.ToolbarSets["Custom"] = [
11:    ['Source'],['Cut','Copy','Paste','PasteText','PasteWord','-',
     'SpellCheck'],
12:  ...
13:  ];
```

这里分别配置了 3 种不同形式的工具栏,FCKConfig. ToolbarSets["Default"]表示默认的工具栏的配置;FCKConfig. ToolbarSets["Basic"]对应于只具备基本功能的工具栏;而 FCKConfig. ToolbarSets["Custom"]则表示自定义的工具栏的配置。一般来说,我们可以通过修改 FCKConfig. ToolbarSets["Custom"]数组的内容,来定义自己的工具栏。该数组的原始设置是与 Default 默认工具栏相同的。数组中的字符串即表示工具栏上的一个按钮,特殊字符'/'表示换行。

若要使生成的在线编辑器使用自定义的工具栏,则应在生成 FCKeditor 的一个对象时,添加工具栏的设置,代码如下:

```
oFCKeditor.ToolbarSet="Custom";
```

一个自定义的工具栏效果如图 8-4 所示,代码可参考 jspbook 项目中的 WebRoot\8\8_2\testFck. jsp。

图 8-4　自定义的 FCKeditor 工具栏

若要使用 FCKeditor 提供的文件上传功能,则需要再下载相应的 jar 文件,并在项目中做相应的配置。

这部分内容请浏览官方网站的相关文档,这里不作详细的介绍。

8.2.3　任务实现

当用户在浏览帖子的内容及回复时,可以在此页面通过点击“回复”链接转向发表回复

的页面，此详情页面的效果如图 8-5 所示。

图 8-5　帖子详情页面

要完成帖子回复的功能，需要编写 2 个页面：

(1)输入回复的页面 newReply.jsp，该页面使用多行文本框或在线编辑器 FCKeditor 提供输入回复内容的功能；

(2)保存回复内容的处理页面 saveReply.jsp，当用户回复的内容不少于指定的字数时，将用户输入的回复标题及内容保存到回复帖表 reply 中。

回复内容输入页面 newReply.jsp 的代码如下所示，该页面使用在线编辑器输入回复内容：

程序清单 文件 WebRoot\8\8_2\newReply.jsp

```
1:  <%@page contentType="text/html" pageEncoding="utf-8"%>
2:  <%
3:      int pid = 0;
4:      if (request.getParameter("pid") != null) {
5:        try {
6:          pid = Integer.parseInt(request.getParameter("pid"));
7:        } catch (Exception e) {
8:          e.printStackTrace(System.err);
9:        }
10:     }
11:     if (pid == 0) {
12:       out.println("<script>history.back();</script>");
13:       return;
14:     }
15:  %>
16:  <! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
17:    "http://www.w3.org/TR/html4/loose.dtd">
18:  <html>
19:    <head>
20:      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```
21:      <title>JSP 学习论坛--发表回复</title>
22:      <link href="css/common.css" rel="stylesheet" type="text/css" />
23:      <script src="js/common.js"></script>
24:      <script type="text/JavaScript"
        src="<%=request.getContextPath()%>/fckeditor/fckeditor.js">
25:      </script>
26:      <script>
27:        function postChk()
28:        {
29:          if ($("ptitle").value==null||$("ptitle").value=="")
30:          {
31:            $("ptitle").value="回复:";
32:          }
33:          var oEditor = FCKeditorAPI.GetInstance("pcontent");
34:          var content = oEditor.GetXHTML(true)
35:          content = content.replace(/<[^<]+>/ig,"");
36:          if (content.sLength()<4)
37:          {
38:            alert("内容太少啦.");
39:            return false;
40:          }
41:          else
42:            return true;
43:        }
44:      </script>
45:    </head>
46:  <body onload="$('ptitle').focus()">
47:   <div id="container">
48:    <%@include file="head.jsp" %>
49:    <div id="content">
50:      <form action="saveReply.jsp" method="post"
              onsubmit="return postChk()">
51:        <input type="hidden" name="pid" value="<%=pid%>"/>
52:        <table style="position:relative">
53:          <caption>请输入回复</caption>
54:            <tr><th>标题</th>
55:    <td>
56:      <input type="text" name="ptitle" id="ptitle" value="" size="90"/>
57:      </td></tr>
58:      <tr><td colspan="2">
59:          <script>
```

```
60:             var sBasePath="<%=request.getContextPath()%>/fckeditor/";
61:             var oFCKeditor = new FCKeditor( 'pcontent' ) ;
62:             oFCKeditor.BasePath  = sBasePath ;
63:             oFCKeditor.Height  = 300 ;
64:             oFCKeditor.ToolbarSet="Custom";
65:             oFCKeditor.Width=700;
66:             oFCKeditor.Value  = '' ;
67:             oFCKeditor.Create() ;
68:           </script>
69:         </td></tr>
70:       <tr>
71:       <td colspan="2" style="text-align:center">
72:       <input type="submit" value="发表" /> 
73:       <input type="button" value="放弃" onclick="history.back()"/>
74:       </td>
75:       </tr>
76:       </table>
77:     </form>
78:   </div> <!--end of content-->
79:   <%@include file="foot.jsp" %>
80:   </div> <!--end of container-->
81:   </body>
82:  </html>
```

■ 代码说明

※ 第 2～15 行代码判断请求参数中是否有相关主题帖的 id，如果没有，则无法继续回复操作，因此通过第 12～13 行代码退回到前一页面，并结束程序；

※ 第 24～25 行引入使用 FCKeditor 所需要的 js 文件；

※ 第 26～44 行自定义了一个 js 函数 postChk，判断所输入的内容的长度是否达到要求；

※ 第 50～77 行提供了一个输入表单，包含 3 项数据：隐藏域 pid，表示主题帖的 id；输入标题的单行文本框 ptitle；第 59～68 行通过 FCKeditor 创建了一个在线编辑框，名称为 pcontent，用以输入回复的内容；

※ 第 50 行的 form 标签表示发生提交事件时，将调用前面编写的 js 函数 postChk 检测所输入的标题和内容的长度，若符合要求，则将表单数据提交到 savePost.jsp 页面进行后继的处理；

当用户填写了回复帖的标题和内容后，将发送给 saveReply.jsp 页面进行处理。该页面接收了来自客户端的数据，而后将其插入到回复帖表 reply 中，具体代码如下：

主要代码 文件 WebRoot\8\8_2\saveReply.jsp

```
<%@page contentType="text/html" pageEncoding="utf-8"
    import="java.util.Date,java.sql.*,java.text.*"%>
//此处略去部分 HTML 标签
<body>
  <%
  request.setCharacterEncoding("utf-8");
  String ptitle, pcontent, pauthor, pid;
  pauthor = (String) session.getAttribute("username");
  ptitle = request.getParameter("ptitle");
  pcontent = request.getParameter("pcontent");
   pid = request.getParameter("pid");
   if (ptitle == null) {
     ptitle = "";
   }
   if (pcontent == null || pcontent.length() < 2 || pid == null) {
     out.print("<script>alert('内容太少啦.');</script>");
     response.sendRedirect("newReply.jsp?pid=" +
         request.getParameter("pid"));
     return;
   }
   String driverClass = "org.gjt.mm.mysql.Driver";
   String url = "jdbc:mysql://localhost:3306/bbs
               ?useUnicode=true&characterEncoding=utf-8";
   String username = "root";
   String password = "rootroot";
   Connection con = null;
   PreparedStatement pt = null;
   try {
     Class.forName(driverClass);
     con = DriverManager.getConnection(url, username, password);
     String sql = "insert into reply(rtitle,rcontent,rauthor,pid)
                  values(?,?,?,?)";
     pt = con.prepareStatement(sql);
     pt.setString(1, ptitle);
     pt.setString(2, pcontent);
     pt.setString(3, pauthor);
     pt.setInt(4, Integer.parseInt(pid));
     pt.executeUpdate();
     SimpleDateFormat sm = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
```

```
36:       Date now = new Date();
37:       pt.executeUpdate("update posts set preplies=preplies+1,plastUpdate='"
              + sm.format(now) + "',plastAuthor='" + pauthor + "' where pid=" + pid);
             //回复数加 1,最后回复时间更新
38:       out.print("<script>alert('回复成功.');</script>");
39:       response.sendRedirect("postDetail.jsp? pid=" + pid);
40:     } catch (Exception e) {
41:       e.printStackTrace(System.err);
42:     } finally {
43:       try {
44:         pt.close();
45:       } catch (Exception e) {
46:         e.printStackTrace(System.err);
47:       }
48:       try {
49:         con.close();
50:       } catch (Exception e) {
51:         e.printStackTrace(System.err);
52:       }
53:     }
54:   %>
55:  </body>
```

■ 代码说明

❖ 第 1 行的 page 指令设置的 import 属性导入所需要的包或类:java.sql 是访问数据库所需要的包,java.util.Date 类用以取得当前的时间,java.text 包中的 SimpleDateFormat 类可将当前时间转换为对应的格式化字符串;

❖ 第 7 行从 session 中获取当前登录用户的用户名,作为发帖的作者;

❖ 第 8～10 行从请求对象 request 中提取了用户提交的回复帖的标题、内容和 ID;

❖ 第 11～18 行检测所输入的内容是否太短,若太少,则在给出提示信息后,重定向到输入回复的页面 newReply.jsp,程序结束;

❖ 第 19～22 行定义了 4 个变量,用以表示连接 MySQL 数据库 bbs 所需要的参数;

❖ 第 23～24 行定义了连接对象 con 和预编译语句对象 pt;

❖ 第 25～54 行完成了记录写入的工作。其中第 26 行加载了驱动类,第 27 行创建了到指定数据库的连接,第 28 行构造了将数据插入表 reply 中的 SQL 语句,其中有 4 个占位符,由 4 个“?”表示,第 29 行从连接对象 con 中创建一个预编译对象并将 SQL 语句装入其中,第 30～33 行设置了对应于 4 个占位符的参数值,第 34 行执行该 SQL 语句;

❖ 除了将记录值写入回复帖表中,还应修改主题帖表中相应记录的回复数、最后作者及

最后发表时间，第 35～37 行完成了该项工作。

页面运行效果如图 8-6 所示：

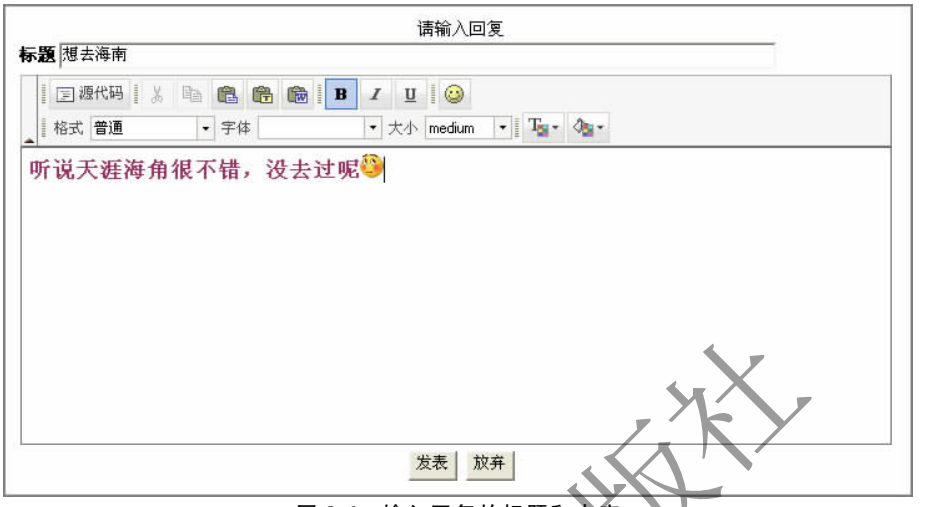

图 8-6　输入回复的标题和内容

回复成功后，将跳转到帖子详情页面，此时可以看到回复的内容已经出现在页面中，如图 8-7 所示：

图 8-7　回复成功

8.2.4　要点提示

■ 在页面中使用 FCKeditor 时，首先需在页面中添加对 fckeditor.js 文件的引用；

■ 可以通过修改配置文件 fckconfig.js，定制自己的工具栏；

■ 创建在线编辑器的方式有 2 种：一是使用 Create 方法直接创建；二是先在页面上创建一个多行文本框，而后使用 FCKeditor 的 ReplaceTextarea 方法以同名的 FCKeditor 对象替换掉该多行文本框。

8.3 数据库通用处理类

前面的章节中，在涉及数据库访问的处理时，采用了直接在页面文件中编写连接代码的形式，这些代码被一次次地重复编写。这种方式除了增加编程的代码量外，还给系统的维护带来了极大的不便，例如：当日后需要修改连接的参数等内容时，需要将所有页面中的此类代码进行修改，工作量之大可想而知。因此，应改为其他更高效易维护的方式。经过分析，可以知道，访问数据库时主要包括以下几种处理：

- 创建数据库连接；
- 执行查询语句；
- 执行增删改或数据定义(DDL)语句；
- 关闭各项数据库资源。

若将这些处理过程封装到一个 Java 类中，则在需要时可直接调用该类完成相应的处理。为了进一步提高代码的通用性，降低代码与使用环境的关联程度，可将连接数据库所需要的参数单独存储在一个属性文件中，连接数据库时从此文件中读取即可，这样，即使连接数据库的参数发生了改变，也不需要重新修改代码。

8.3.1 编写通用处理类

创建数据库连接时，需要指明连接数据库的驱动类名称、连接 url、用户名以及密码，我们将这些参数保存在属性文件 dbconn.properties 中，并存放在项目的 src 目录下，文件内容如下：

文件内容 src\dbconn.properties

```
driver =com.mysql.jdbc.Driver
url =jdbc:mysql://localhost:3306/bbs? useUnicode=true&characterEncoding=utf8
username = root
password =rootroot
```

这里连接的是 MySQL 数据库，数据库名称为“bbs”，连接用户名是“root”，连接为“rootroot”，实际使用时应根据具体情况做相应的更改。

通用处理类 Conn.java 的代码如下所示：

程序清单 src\dbUtil\Conn.java

```
1:  package dbUtil;
2:
3:  import java.io.InputStream;
4:  import java.sql.*;
5:  import java.util.Properties;
6:
```

```
7:  public class Conn {
8:      private Connection con;
9:      private Statement st;
10:  //构造函数,在创建 Conn 类的对象时自动调用
11:      public Conn() {
12:      try{
13:          Properties pros = new Properties();
14:          InputStream in;
15:          in= Conn.class.getResourceAsStream("/dbconn.properties");
16:          pros.load(in);
17:          in.close();
18:          String driver = pros.getProperty("driver");
19:          String url = pros.getProperty("url");
20:          String username = pros.getProperty("username");
21:          String password = pros.getProperty("password");
22:          Class.forName(driver);
23:          con = DriverManager.getConnection(url,username,password);
24:          st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                   ResultSet.CONCUR_UPDATABLE);
25:      }catch(Exception e){}
26:      }
27:  //executeQuery--执行查询语句
28:      public ResultSet executeQuery(String sql) {
29:          ResultSet rs = null;
30:          try {
31:              rs = st.executeQuery(sql);
32:          } catch (Exception e) {
33:              e.printStackTrace(System.err);
34:          }
35:          return rs;
36:      }
37:   //executeUpdate 方法--执行对数据库的更新的 SQL 语句
38:      public int executeUpdate(String sql) {
39:          try {
40:              return st.executeUpdate(sql);
41:          } catch (Exception e) {
42:               e.printStackTrace(System.err);
43:              return 0;
44:          }
45:      }
46:  //创建预编译对象的方法
```

```
47:      public PreparedStatement prepareStatement(String sql) {
48:          try {
49:              return con.prepareStatement(sql);
50:          } catch (SQLException e) {
51:              e.printStackTrace(System.err);
52:              return null;
53:          }
54:      }
55:  //关闭各项资源的方法
56:      public void close() {
57:          try {
58:              if (st! =null) st.close();
59:          } catch (Exception e) {
60:              e.printStackTrace(System.err);
61:          }
62:          try {
63:              if (con! =null) con.close();
64:          } catch (Exception e) {
65:             e.printStackTrace(System.err);
66:          }
67:      }
68:  }
```

■ 代码说明

※ 该类共有 5 个方法：构造方法 Conn、执行 select 语句的方法 executeQuery、执行 insert、delete、update 等 SQL 语句的方法 executeUpdate、返回一个预编译的 PreparedStatement 对象的方法 prepareStatement 以及逆序关闭各项资源的方法 close；

※ 在构造方法中通过第 13～21 行读取了存放在 src 目录下的属性文件 dbconn.properties 中的 4 个属性，第 22 行加载了驱动类，第 23 行获取数据库连接，第 24 行创建了一个 Statement 对象。

8.3.2 使用通用处理类

编写了通用处理类后，其他页面均可通过调用该类来完成数据库访问。这里以前面第 6.6 节中已经完成的注册功能为例，说明如何在程序中使用数据库通用处理类。

修改原 6.6 节中的 chkReg.jsp 页面，在 page 指令中使用 import 属性添加对 dbUtil.Conn 类的导入，而后将其中 if (msg.equals("")){...}部分的代码修改为：

部分代码 WebRoot\8\8_3\chkReg.jsp

```
1:  if (msg.equals("")) {
2:    Conn conn = null;//声明连接对象
3:    ResultSet rs = null;   // 声明结果集对象
```

```
4:    try {
5:      conn = new Conn(); //获取连接
6:      String sql = "select * from users where uname='" + uname + "'";
7:      rs = conn.executeQuery(sql);
8:      if (rs != null) {
9:        if (rs.next()) {
10:         msg = "用户名<b><font color='red'> " + uname +
                " </font></b>已被使用,请改用其他名字.";
11:       }
12:     }
13:     if (msg.equals("")) {//用户名可用,执行插入操作
14:       sql = "insert into users(uname,upw,usex,uemail,ulogo)
                  values('" + uname + "'";
15:       sql += ",'" + upw + "','" + usex + "','" + uemail + "','" +
            ulogo + "')";
16:       int t = conn.executeUpdate(sql);
17:       if (t > 0) {
18:         msg = "恭喜! 注册成功!";
19:       } else {
20:         msg = "Sorry,注册失败.";
21:       }
22:     }
23:   } catch (Exception e) {
24:     out.println("网站发生异常.");
25:     e.printStackTrace(System.err);
26:   } finally {
27:     try {
28:       if (rs != null) {
29:         rs.close();
30:       }
31:     } catch (Exception e) {
32:     }
33:     try {
34:       if (con != null) {
35:         conn.close();
36:       }
37:     } catch (Exception e) {
38:     }
39:   }
40: }
```

■ 代码说明

※ 上面的代码段中，第 2 行和第 5 行声明并创建了一个 Conn 类的对象 conn；

※ 第 7 行通过 conn 对象调用 Conn 类的 executeQuery 方法直接执行 select 语句，以查询用户信息表中是否存在与指定用户名相同的记录；

※ 第 16 行通过 conn 对象调用 Conn 类的 executeUpdate 方法执行 insert 语句，将新用户的注册信息插入到用户信息表中；

※ 第 34～36 行通过 conn 对象调用了 Conn 类的 close 方法，该方法的作用是关闭所创建的连接对象和 Statement 对象；

※ 应注意：在此段代码中，并没有直接创建 Statement 对象，也没有直接使用该类型的对象去执行 SQL 语句，所有执行 SQL 语句的工作都是交给 Conn 类的相关方法去完成的。

请读者按照上述的方法，修改论坛网站的其他功能，将其中通过 JDBC 对数据库的访问改用使用通用类 Conn 实现。

【专项训练】

● 训练一

【训练目标】

练习在线编辑器 FCKeditor 的使用方法。

【训练任务】

在页面 inputMessage.jsp 使用 FCKeditor 提供意见输入框，提交到 printMessage.jsp 后，在此页面显示用户所输入的意见信息。

● 训练二

【训练目标】

使用数据通用处理类封装对数据库的一般访问。

【训练任务】

使用通用处理类实现对数据库的访问，完成论坛发表新帖功能。

【学生项目实施】

● 初级任务——网络点餐系统的数据库处理程序

➢ 编写类 jdbc.DBBean，实现连接数据库时的驱动程序加载和各种对象的创建。

➢ 该类提供执行查询和更新操作的方法以及关闭对象和数据库连接的方法。

● 初级任务——网络点餐系统的管理员添加留言功能

➢ 编写页面 addMessage.jsp(图 4-34)，提供留言标题输入框和留言内容的在线编辑区域，该页面提交到 addMessage_do.jsp。

➢ addMessage_do.jsp 读取留言标题和内容并保存到数据库。

● 中级任务——网络点餐系统的用户留言功能

➢ 编写页面 messageBoard.jsp(图 4-21),读取系统的所有留言信息并显示,每页显示 2 条留言,同时提供分页功能。

➢ 页面的下方是发表留言的区域,包含留言标题输入框和留言内容的在线编辑区域。

➢ 留言提交到 addMessage.jsp 页面,该页面读取留言标题和内容保存到数据库,并重定向到 messageBoard.jsp 页面。

● 高级任务——网络点餐系统的管理员留言功能

➢ 编写页面 messageBoard.jsp(图 4-32),读取并显示系统中的所有留言信息(最新留言显示在前面),每页显示 5 条留言,同时提供分页功能。每条留言信息的后面提供一个回复链接和一个删除链接。

➢ 回复链接的地址为 replyMessage.jsp(图 4-34),显示“Re:”加用户留言标题,并提供回复内容的在线编辑区域,该页面提交到 replyMessage_do.jsp。

➢ replyMessage_do.jsp 页面负责读取回复内容并保存到数据库。

➢ 删除链接的地址为 deleteMessage.jsp,当点击删除链接时,会弹出询问对话框(图 4-34),点击“是”后,将会删除本条留言。