

清爽夏日九宫格日记网

(Java Web+Ajax+JQuery+MySQL 实现)

随着工作和生活节奏的不断加快，属于自己的私人时间也越来越少，日记这种传统的倾诉方式也逐渐被人们所淡忘，取而代之的是各种各样的网络日志。不过，最近网络中又出现了一种全新的日记方式，九宫格日记，它由九个方方正正的格子组成，让用户可以像做填空题那样对号入座，填写相应的内容，从而完成一篇日记，整个过程不过几分钟。九宫格日记因其便捷、省时等优点在网上迅速风行开来，倍受学生、年轻上班族所青睐。目前很多公司白领也在写九宫格日记。本章将以清爽夏日九宫格日记网为例介绍如何应用 Java Web+Ajax+JQuery+MySQL 实现九宫格日记网。

通过阅读本章，读者可以学到：

- » 应用 DIV+CSS 进行网站布局
- » 实现 Ajax 重构
- » 图片的展开和收缩的方法
- » 如何进行图片的左转和右转
- » 查看日记图片原图的技术
- » 如何根据指定内容生成 PNG 图片
- » 生成图片缩略图的技术
- » 如何弹出灰色半透明背景的无边框窗口
- » 根据文字的个数控制输出文字的尺寸
- » 应用 JQuery 让 PNG 图片在 IE 6 下背景透明
- » 配置解决中文乱码的过滤器

A.1 项目设计思路

A.1.1 功能阐述

清爽夏日九宫格日记网主要包括显示九宫格日记列表、写九宫格日记和用户等 3 个功能模块。下面分别进行介绍。

显示九宫格日记列表主要用于分页显示全部九宫格日记、分页显示我的日记、展开和收缩日记图片、显示日记原图、对日记图片进行左转和右转以及删除自己的日记等。

写九宫格日记主要用于填写日记信息、预览生成的日记图片和保存日记图片。其中，在填写日记信息时，允许用户选择并预览自己喜欢的模板，以及选择预置日记内容等。

用户模块又包括以用户注册、用户登录、退出登录和找回密码等 4 个子功能，下面分别进行说明。

- ☒ 用户注册主要用于新用户注册。在进行用户注册，系统将采用 Ajax 实现无刷新的验证和保存注册信息。
- ☒ 用户登录主要用于用户登录网站，登录后的用户可以查看自己的日记、删除自己的日记以及写九宫格日记等。
- ☒ 退出登录主要用于登录用户退出当前登录状态。
- ☒ 找回密码主要用于当用户忘记密码时，根据密码提示问题和答案找回密码。

A.1.2 系统预览

为了让读者对本系统有个初步的了解和认识，下面给出本系统的几个页面运行效果图。

分页显示九宫格日记列表如图 A.1 所示，该页面用于分页显示日记列表，包括展开和收缩日记图片、显示日记原图、对日记图片进行左转和右转等功能。当用户登录后，还可以查看和删除自己的日记。



图 A.1 分页显示九宫格日记列表页面

写九宫格日记页面如图 A.2 所示, 该页面用于填写日记信息, 允许用户选择并预览自己喜欢的模板, 以及选择预置日记内容等。

预览九宫格日记页面如图 A.3 所示，该页面主要用于预览日记图片，如果用户满意，可以单击“保存”超级链接保存日记图片，否则可以单击“再改改”超级链接返回填写九宫格日记页面进行修改。

标题：无题

开心的事 工作完成了	为他人做的事 问候家人了	工作/计划/备忘 继续努力工作
比起昨天的进步 效率提高了	2011年6月7日 星期二 	心情/感悟/灵感 神马都是浮云
关注/新闻/八卦 她/他写九宫格日记了	健康/体重/饮食 一日三餐不能少	梦/梦想 睡得很好


再改改 保存

图 A.3 预览九宫格日记页面

用户注册页面如图 A.4 所示，该页面用于实现用户注册。在该页面中，输入用户名后，将光标移出该文本框，系统将自动检测输入的用户名是否合法（包括用户名长度及是否被注册），如果不合法，将给出错误提示，同样，输入其他信息时，系统也将实时检测输入的信息是否合法。

清爽夏日九宫格日记网-用户注册

很抱歉，[mr]已经被注册！

用户名: *长度限制为20个字母或10个汉字

密码: *密码由字母开头的字母、数字或下划线组成，并且密码的长度大于6位
小于30位

确认密码: *请确认密码

E-mail地址: *请输入有效的E-mail地址，在找回密码时应用

所在地: -

以下两个选项，只要有任何一个没有输入，将不可以通过答案问题重新设置密码。

密码提示问题: 如: 我的工作单位

提示问题答案: 如: 明日科技

图 A.4 用户注册页面

A.1.3 功能结构

清爽夏日九宫格日记网的功能结构如图 A.5 所示。

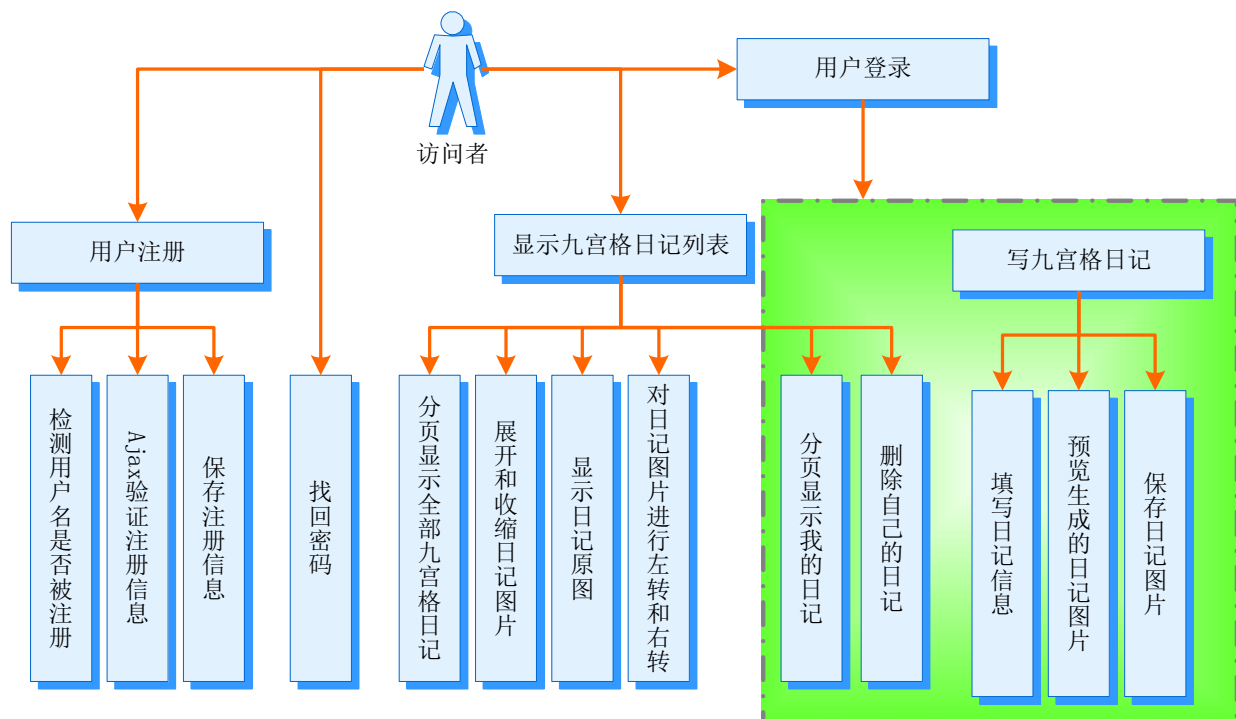



图 A.5 清爽夏日九宫格日记网的功能结构图

 说明：在图 A.5 中，用虚线框起来的部分为只有用户登录后，才可以有的功能。

A.1.4 文件夹组织结构

在进行清爽夏日九宫格日记网开发之前，要对系统整体文件夹组织架构进行规划。对系统中使用的文件进行合理的分类，分别放置于不同的文件夹下。通过对文件夹组织架构的规划，可以确保系统文件目录明确、条理清晰，同样也便于系统的更新和维护。本项目的文件夹组织架构规划如图 A.6 所示。

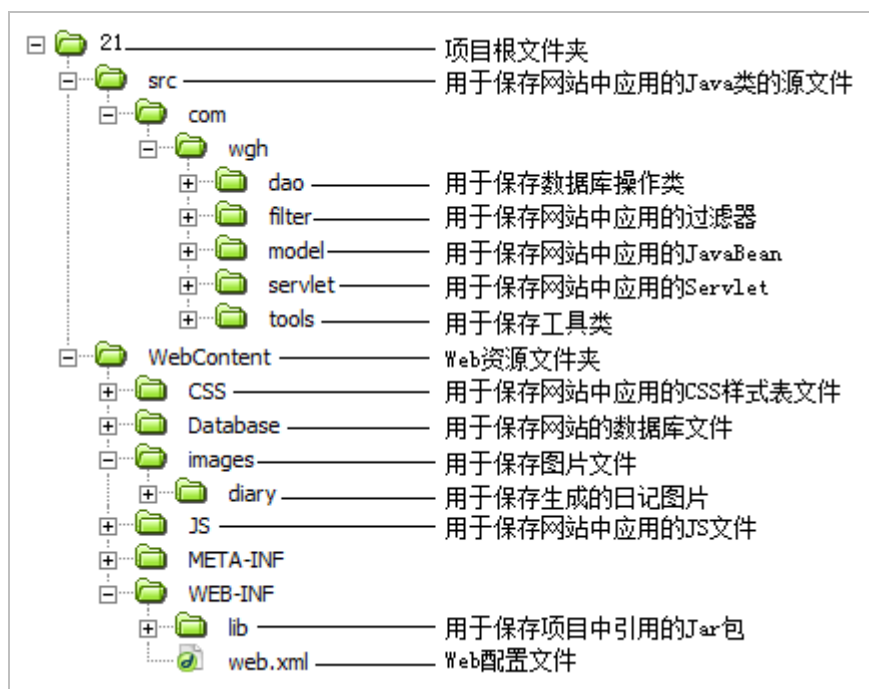


图 A.6 清爽夏日九宫格日记网的文件夹组织结构

A.2 数据库设计

A.2.1 数据库设计

结合实际情况及对功能的分析，规划清爽夏日九宫格日记网的数据库，定义数据库名称为 db_9griddiary，数据库主要包含两张数据表，如图 A.7 所示。

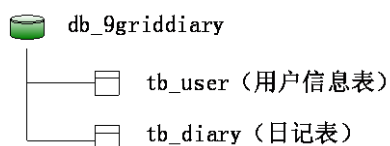


图 A.7 清爽夏日九宫格日记网的数据库

A.2.2 数据表设计

清爽夏日九宫格日记网的数据库中包括两张数据表，如 A.1 表和表 A.2 所示

1. tb_user（用户信息表）

用户信息表主要用于存储用户的注册信息。该数据表的结构如表 A.1 所示。

表 A.1 tb_user 表


字 段 名 称	数 据 类 型	字 段 大 小	是 否 主 键	说 明
id	INT	10	主键	自动编号 ID
username	VARCHAR	50		用户名
pwd	VARCHAR	50		密码
email	VARCHAR	100		E-mail
question	VARCHAR	45		密码提示问题
answer	VARCHAR	45		提示问题答案
city	VARCHAR	30		所在地

2. tb_diary（日记表）

日记表主要用于存储日记的相关信息。该数据表的结构如图 A.2 所示。

表 A.2 tb_diary 表

字 段 名 称	数 据 类 型	字 段 大 小	是 否 主 键	说 明
id	INT	10	主键	自动编号 ID
title	VARCHAR	60		标题
address	VARCHAR	50		日记保存的地址
writeTime	TIMESTAMP			写日记时间
userid	INT	10		用户 ID

 说明：在设置数据表 tb_diary 时，还需要为字段 writeTime 设置默认值，这里为 CURRENT_TIMESTAMP，也就是当前时间。

A.3 公共模块设计


在开发过程中，经常会用到一些公共模块，例如，数据库连接及操作的类、保存分页代码的 JavaBean、解决中文乱码的过滤器及实体类等。因此，在开发系统前首先需要设计这些公共模块。下面将具体介绍清爽夏日九宫格日记网所需要的公共模块的设计过程。

A.3.1 编写数据库连接及操作的类

数据库连接及操作类通常包括连接数据库的方法 getConnection()、执行查询语句的方法 executeQuery()、执行更新操作的方法 executeUpdate()、关闭数据库连接的方法 close()。下面将详细介绍如何编写清爽夏日九宫格日记网的数据库连接及操作的类 ConnDB。

(1) 指定类 ConnDB 保存的包，并导入所需的类包，本例将其保存到 com.wgh.tools 包中，代码如下：

```
package com.wgh.tools;           //将该类保存到com.wgh.tools包中
import java.io.InputStream;      //导入java.io.InputStream类
import java.sql.*;               //导入java.sql包中的所有类
import java.util.Properties;     //导入java.util.Properties类
```


 **注意：**包语句以关键字 `package` 后面紧跟一个包名称，然后以分号“;”结束；包语句必须出现在 `import` 语句之前；一个 `.java` 文件只能有一个包语句。

(2) 定义 `ConnDB` 类，并定义该类中所需的全局变量及构造方法，代码如下：

```
public class ConnDB {
    public Connection conn = null;           // 声明Connection对象的实例
    public Statement stmt = null;           // 声明Statement对象的实例
    public ResultSet rs = null;             // 声明ResultSet对象的实例
    private static String propFileName = "connDB.properties"; // 指定资源文件保存的位置
    private static Properties prop = new Properties(); // 创建并实例化Properties对象的实例
    private static String dbClassName = "com.mysql.jdbc.Driver"; // 定义保存数据库驱动名的变量
    private static String dbUrl = "jdbc:mysql://127.0.0.1:3306/db_9griddiary?
    user=root&password=111&useUnicode=true";
    public ConnDB() {                       // 构造方法
        try {                               // 捕捉异常
            // 将Properties文件读取到InputStream对象中
            InputStream in = getClass().getResourceAsStream(propFileName);
            prop.load(in);                   // 通过输入流对象加载Properties文件
            dbClassName = prop.getProperty("DB_CLASS_NAME"); // 获取数据库驱动
            // 获取连接的URL
            dbUrl = prop.getProperty("DB_URL", dbUrl);
        } catch (Exception e) {
            e.printStackTrace();             // 输出异常信息
        }
    }
}
```

(3) 为了方便程序移植，这里将数据库连接所需信息保存到 `properties` 文件中，并将该文件保存在 `com.wgh.tools` 包中。`connDB.properties` 文件的内容如下：

```
DB_CLASS_NAME=com.mysql.jdbc.Driver
DB_URL=jdbc:mysql://127.0.0.1:3306/db_9griddiary?user=root&password=111&useUnicode=true
```

 **说明：**`properties` 文件为本地资源文本文件，以“消息/消息文本”的格式存放数据。使用 `Properties` 对象时，首先需创建并实例化该对象，代码如下：

```
private static Properties prop = new Properties();
再通过文件输入流对象加载 Properties 文件，代码如下：
prop.load(new FileInputStream(propFileName));
最后通过 Properties 对象的 getProperty 方法读取 properties 文件中的数据。
```

(4) 创建连接数据库的方法 `getConnection()`，该方法返回 `Connection` 对象的一个实例。`getConnection()` 方法的代码如下：

```

public static Connection getConnection() {
    Connection conn = null;
    try {
        Class.forName(dbClassName).newInstance(); // 连接数据库时可能发生异常因此需要捕捉该异常
        conn = DriverManager.getConnection(dbUrl); // 装载数据库驱动
    } catch (Exception ee) { // 建立与数据库URL中定义的数据库的连接
        ee.printStackTrace(); // 输出异常信息
    }
    if (conn == null) {
        System.err.println("警告: DbConnectionManager.getConnection() 获得数据库链接失败.\r\n链接类型:"
            + dbClassName + "\r\n链接位置:" + dbUrl); // 在控制台上输出提示信息
    }
    return conn; // 返回数据库连接对象
}

```

关键代码解析

04 该句代码用于利用 Class 类中的静态方法 forName(), 加载要使用的 Driver。使用该语句可以将传入的 Driver 类名称的字符串当作一个 Class 对象, 通过 newInstance()方法可以建立此 Class 对象的一个新实例。

05 DriverManager 用于管理 JDBC 驱动程序的接口, 通过其 getConnection()方法来获取 Connection 对象的引用。Connection 对象的常用方法如下。

Statement createStatement(): 创建一个 Statement 对象, 用于执行 SQL 语句。

close(): 关闭数据库的连接, 在使用完连接后必须关闭, 否则连接会保持一段比较长的时间, 直到超时。

PreparedStatement prepareStatement(String sql): 使用指定的 SQL 语句创建了一个预处理语句, sql 参数中往往包含一个或多个 “?” 占位符。

CallableStatement prepareCall(String sql): 创建一个 CallableStatement 用于执行存储过程, sql 参数是调用的存储过程, 中间至少包含一个 “?” 占位符。

(5) 创建执行查询语句的方法 executeQuery, 返回值为 ResultSet 结果集。executeQuery 方法的代码如下:

```

public ResultSet executeQuery(String sql) {
    try {
        conn = getConnection(); // 捕捉异常
        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, // 调用getConnection()方法构造Connection对象的一个实例conn
            ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery(sql);
    } catch (SQLException ex) {
        System.err.println(ex.getMessage()); // 输出异常信息
    }
    return rs; // 返回结果集对象
}

```

关键代码解析

07 ResultSet.TYPE_SCROLL_INSENSITIVE 常量允许记录指针向前或向后移动, 且当 ResultSet 对象变动记录指针时, 会影响记录指针的位置。

08 ResultSet.CONCUR_READ_ONLY 常量可以解释为 ResultSet 对象仅能读取, 不能修改, 在对数据库的查询操作中使用。

09 stmt 为 Statement 对象的一个实例, 通过其 executeQuery(String sql)方法可以返回一个 ResultSet 对象。

(6) 创建执行更新操作的方法 `executeUpdate()`，返回值为 `int` 型的整数，代表更新的行数。`executeQuery()`方法的代码如下：

```
public int executeUpdate(String sql) {
    int result = 0;                // 定义保存返回值的变量
    try {                          // 捕捉异常
        conn = getConnection();    // 调用getConnection()方法构造Connection对象的一个实例conn
        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                     ResultSet.CONCUR_READ_ONLY);
        result = stmt.executeUpdate(sql); // 执行更新操作
    } catch (SQLException ex) {
        result = 0;                // 将保存返回值的变量赋值为0
    }
    return result;                // 返回保存返回值的变量
}
```

(7) 创建关闭数据库连接的方法 `close()`。`close()`方法的代码如下：

```
public void close() {
    try {                          // 捕捉异常
        if (rs != null) {         // 当ResultSet对象的实例rs不为空时
            rs.close();           // 关闭ResultSet对象
        }
        if (stmt != null) {       // 当Statement对象的实例stmt不为空时
            stmt.close();         // 关闭Statement对象
        }
        if (conn != null) {       // 当Connection对象的实例conn不为空时
            conn.close();         // 关闭Connection对象
        }
    } catch (Exception e) {
        e.printStackTrace(System.err); // 输出异常信息
    }
}
```

A.3.2 编写保存分页代码的 JavaBean

由于在清爽夏日九宫格日记网中，需要对日记列表进行分页显示，所以需要编写一个保存分页代码的 JavaBean。保存分页代码的 JavaBean 的具体编写步骤如下。

(1) 编写用于保存分页代码的 JavaBean，名称为 `MyPagination`，保存在“com.wgh.tools”包中，并定义一个全局变量 `list` 和 3 个局部变量，关键代码如下：

```
public class MyPagination {
    public List<Diary> list=null;
    private int recordCount=0;    //保存记录总数的变量
    private int pagesize=0;      //保存每页显示的记录数的变量
    private int maxPage=0;       //保存最大页数的变量
}
```

(2) 在 JavaBean “MyPagination” 中添加一个用于初始化分页信息的方法 `getInitPage()`，该方法包括 3 个参数，分别是用于保存查询结果的 List 对象 `list`、用于指定当前页面的 `int` 型变量 `Page` 和用于指定每页显示的记录数的 `int` 型变量 `pagesize`。该方法的返回值为保存要显示记录的 List 对象。具体代码如下：

```
public List<Diary> getInitPage(List<Diary> list,int Page,int pagesize){
    List<Diary> newList=new ArrayList<Diary>();
    this.list=list;
    recordCount=list.size();                //获取list集合的元素个数
    this.pagesize=pagesize;
    this.maxPage=getMaxPage();             //获取最大页数
    try{                                    //捕获异常信息
        for(int i=(Page-1)*pagesize;i<=Page*pagesize-1;i++){
            try{
                if(i>=recordCount){break;}    //跳出循环
            }catch(Exception e){}
            newList.add((Diary)list.get(i));
        }
    }catch(Exception e){
        e.printStackTrace();                //输出异常信息
    }
    return newList;
}
```

(3) 在 JavaBean “MyPagination” 中添加一个用于获取指定页数据的方法 `getAppointPage()`，该方法只包括一个用于指定当前页数的 `int` 型变量 `Page`。该方法的返回值为保存要显示记录的 List 对象。具体代码如下：

```
public List<Diary> getAppointPage(int Page){                //获取指定页的数据
    List<Diary> newList=new ArrayList<Diary>();
    try{
        for(int i=(Page-1)*pagesize;i<=Page*pagesize-1;i++){    //通过for循环获取当前页的数据
            try{
                if(i>=recordCount){break;}                //跳出循环
            }catch(Exception e){}
            newList.add((Diary)list.get(i));
        }
    }catch(Exception e){
        e.printStackTrace();                //输出异常信息
    }
    return newList;
}
```

(4) 在 JavaBean “MyPagination” 中添加一个用于获取最大记录数的方法 `getMaxPage()`，该方法无参数，其返回值为最大记录数。具体代码如下：

```
public int getMaxPage(){
    int maxPage=(recordCount%pagesize==0)?(recordCount/pagesize):(recordCount/pagesize+1);
}
```

```
        return maxPage;
    }
}
```

(5) 在 JavaBean “MyPagination” 中添加一个用于获取总记录数的方法 `getRecordSize()`，该方法无参数，其返回值为总记录数。具体代码如下：

```
public int getRecordSize(){
    return recordCount;
}
```

(6) 在 JavaBean “MyPagination” 中添加一个用于获取当前页数的方法 `getPage()`，该方法只有一个用于指定从页面中获取的页数的参数，其返回值为处理后的页数。具体代码如下：

```
public int getPage(String str){
    if(str==null){                                //当页数等于null时，让其等于0
        str="0";
    }
    int Page=Integer.parseInt(str);
    if(Page<1){                                    //当页数小于1时，让其等于1
        Page=1;
    }else{
        if(((Page-1)*pagesize+1)>recordCount){    //当页数大于最大页数时，让其等于最大页数
            Page=maxPage;
        }
    }
    return Page;
}
```

(7) 在 JavaBean “MyPagination” 中添加一个用于输出记录导航的方法 `printCtrl()`，该方法包括 3 个参数，分别为 `int` 型的 `Page`（当前页数）、`String` 类型的 `url`（URL 地址）和 `String` 类型的 `para`（要传递的参数），其返回值为输出记录导航的字符串。具体代码如下：

```
public String printCtrl(int Page,String url,String para){
    String strHtml="<table width='100%' border='0' cellspacing='0' cellpadding='0'><tr> <td height='24' align='right'>当前页数： 【"+Page+"/"+maxPage+"】&nbsp;";
    try{
        if(Page>1){
            strHtml=strHtml+"<a href='"+url+"&Page=1"+para+"'>第一页</a> ";
            strHtml=strHtml+"<a href='"+url+"&Page="+Page+"'+>上一页</a>";
        }
        if(Page<maxPage){
            strHtml=strHtml+"<a href='"+url+"&Page="+Page+"'+>下一页</a>";
            strHtml=strHtml+"<a href='"+url+"&Page="+maxPage+para+"'>最后一页&nbsp;";
        }
        strHtml=strHtml+"</td> </tr> </table>";
    }catch(Exception e){
        e.printStackTrace();
    }
    return strHtml;
}
```

A.3.3 配置解决中文乱码的过滤器

在程序开发时，通常有两种方法解决程序中经常出现的中文乱码问题，一种是通过编码字符串处理类，对需要的内容进行转码；另一种是配置过滤器。其中，第二种方法比较方便，只需要在开发程序时配置正确即可。下面将介绍本系统中配置解决中文乱码的过滤器的具体步骤。

(1)编写 `CharacterEncodingFilter` 类,让它实现 `Filter` 接口,成为一个 `Servlet` 过滤器,在实现 `doFilter()` 接口方法时,根据配置文件中设置的编码格式参数分别设置请求对象的编码格式和响应对象的内容类型参数。

```
public class CharacterEncodingFilter implements Filter {
    protected String encoding = null;           // 定义编码格式变量
    protected FilterConfig filterConfig = null; // 定义过滤器配置对象
    public void init(FilterConfig filterConfig) throws ServletException {
        this.filterConfig = filterConfig;       // 初始化过滤器配置对象
        this.encoding = filterConfig.getInitParameter("encoding"); // 获取配置文件中指定的编码格式
    }
    // 过滤器的接口方法，用于执行过滤业务
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        if (encoding != null) {
            request.setCharacterEncoding(encoding); // 设置请求的编码
            // 设置响应对象的内容类型（包括编码格式）
            response.setContentType("text/html; charset=" + encoding);
        }
        chain.doFilter(request, response); // 传递给下一个过滤器
    }
    public void destroy() {
        this.encoding = null;
        this.filterConfig = null;
    }
}
```

(2) 在 `web-inf.xml` 文件中配置过滤器，并设置编码格式参数和过滤器的 `URL` 映射信息。关键代码如下。

```
<filter>
  <filter-name>CharacterEncodingFilter</filter-name>      <!--指定过滤器类文件-->
  <filter-class>com.wgh.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>                      <!--指定编码为UTF-8编码-->
  </init-param>
</filter>
<filter-mapping>
  <filter-name>CharacterEncodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
  <!--设置过滤器对应的请求方式-->
</filter-mapping>
```

```
<dispatcher>REQUEST</dispatcher>
<dispatcher>FORWARD</dispatcher>
</filter-mapping>
```

A.3.4 编写实体类

实体类就是由属性及属性所对应的 `getter` 和 `setter` 方法组成的类。实体类通常与数据表相关联。在清爽夏日九宫格日记网中，共涉及到两张数据表，分别是用户信息表和日记表。通过这两张数据表可以得到用户信息和日记信息，根据这些信息可以得出用户实体类和日记实体类。由于实体类的编写方法基本类似，所以这里将以日记实体类为例进行介绍。

(1) 编写 `Diary` 类，在该类添加 `id`、`title`、`address`、`writeTime`、`userid` 和 `username` 属性，并为这些属性添加对应的 `getter` 和 `setter` 方法，关键代码如下：

```
import java.util.Date;
public class Diary {
    private int id = 0;           // 日记ID号
    private String title = "";    // 日记标题
    private String address = "";  // 日记图片地址
    private Date writeTime = null; // 写日记的时间
    private int userid = 0;       // 用户ID
    private String username = ""; // 用户名
    public int getId() {          // id属性对应的getter方法
        return id;
    }

    public void setId(int id) {   // id属性对应的setter方法
        this.id = id;
    }
    //此处省略了其他属性对应的getter和setter方法
}
```

A.4 主界面设计

A.4.1 主界面概述

当用户访问清爽夏日九宫格日记网时，首先进入的是网站的主界面。清爽夏日九宫格日记网的主界面主要包括以下 4 部分内容：

- ☑ **Banner 信息栏：**主要用于显示网站的 Logo。
- ☑ **导航栏：**主要用于显示网站的导航信息及欢迎信息。其中导航条目将根据是否登录而显示不同的内容。
- ☑ **主显示区：**主要用于分页显示九宫格日记列表。
- ☑ **版权信息栏：**主要用于显示版权信息。

下面看一下本项目中设计的主界面，如图 A.8 所示。



图 A.8 清爽夏日九宫格日记网的主界面

A.4.2 让采用 DIV+CSS 布局的页面内容居中

清爽夏日九宫格日记网采用 DIV+CSS 布局。在采用 DIV+CSS 布局的网站中，一个首要问题就是如何让页面内容居中。下面将介绍具体的实现方法。

(1) 在页面的<body>标记的下方添加一个<div>标记（使用该<div>标记将页面内容括起来），并设置其 id 属性，这里将其设置为 box，关键代码如下：

```
<div id="box">
  <!--页面内容-->
</div>
```

(2) 设置 CSS 样式。这里通过在链接的外部样式表文件中进行设置。

```
body{
  margin:0px;          /*设置外边距*/
  padding:0px;         /*设置内边距*/
  font-size: 9pt;      /*设置字体大小*/
}
```

```
#box{
    margin:0 auto auto auto;    /*设置外边距*/
    width:800px;                /*设置页面宽度*/
    clear:both;                 /*设置两侧均不可以有浮动内容*/
    background-color: #FFFFFF; /*设置背景颜色*/
}
```

 **注意：**在 JSP 页面中，一定要包含以下代码，否则页面内容将不居中。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

A.4.3 主界面的实现过程

在清爽夏日九宫格日记网主界面中，Banner 信息栏、导航栏和版权信息，并不是仅存在于主界面中，其他功能模块的子界面中也需要包括这些部分。因此，可以将这几个部分分别保存在单独的文件中，这样，在需要放置相应功能时只需包含这些文件即可。

在 JSP 页面中包含文件有两种方法：一种是应用 `<%@ include %>` 指令实现，另一种是应用 `<jsp:include>` 动作元素实现。

`<%@ include %>` 指令用来在 JSP 页面中包含另一个文件。包含的过程是静态的，即在指定文件属性值时，只能是一个包含相对路径的文件名，而不能是一个变量，也不可以在所指定的文件后面添加任何参数。其语法格式如下：

```
<%@ include file="fileName"%>
```

`<jsp:include>` 动作元素可以指定加载一个静态或动态的文件，但运行结果不同。如果指定为静态文件，那么这种指定仅仅是把指定的文件内容加到 JSP 文件中，则这个文件不被编译。如果是动态文件，那么这个文件将会被编译器执行。由于在页面中包含查询模块时，只需要将文件内容添加到指定的 JSP 文件中即可，所以此处可以使用加载静态文件的方法包含文件。应用 `<jsp:include>` 动作元素加载静态文件的语法格式如下：

```
<jsp:include page="{relativeURL | <%=expression%>}" flush="true"/>
```

使用 `<%@ include %>` 指令和 `<jsp:include>` 动作元素包含文件的区别是：使用 `<%@ include %>` 指令包含的页面，是在编译阶段将该页面的代码插入到了主页面的代码中，最终包含页面与被包含页面生成了一个文件。因此，如果被包含页面的内容有改动，需重新编译该文件。而使用 `<jsp:include>` 动作元素包含的页面可以是动态改变的，它是在 JSP 文件运行过程中被确定的，程序执行的是两个不同的页面，即在主页面中声明的变量，在被包含的页面中是不可见的。由此可见，当被包含的 JSP 页面中包含动态代码时，为了不和主页面中的代码相冲突，需要使用 `<jsp:include>` 动作元素包含文件。应用 `<jsp:include>` 动作元素包含查询页面的代码如下：

```
<jsp:include page="search.jsp" flush="true"/>
```

考虑到本系统中需要包含的多个文件之间相对比较独立，并且不需要进行参数传递，属于静态包含，因此采用 `<%@ include %>` 指令实现。

应用 `<%@ include %>` 指令包含文件的方法进行主界面布局的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>显示九宫格日记列表</title>
</head>
<body bgcolor="#F0F0F0">
  <div id="box">
    <% @ include file="top.jsp" %>
    <% @ include file="register.jsp" %>
    <!--显示九宫格日记列表的代码-->
    <% @ include file="bottom.jsp" %>
  </div>
</body>
</html>
```

A.5 用户模块设计

A.5.1 用户模块概述

在清爽夏日九宫格日记网中，如果用户没有注册为网站的用户，则只能浏览他人的九宫格日记。如果想要发表自己的日记，则要注册为网站的用户。当用户成功注册为网站的用户后，就可以登录到本网站，并发表自己的日记，当用户忘记自己的密码时，还可以通过“找回密码”功能找回自己的密码。用户注册模块的运行结果如图 A.9 所示。

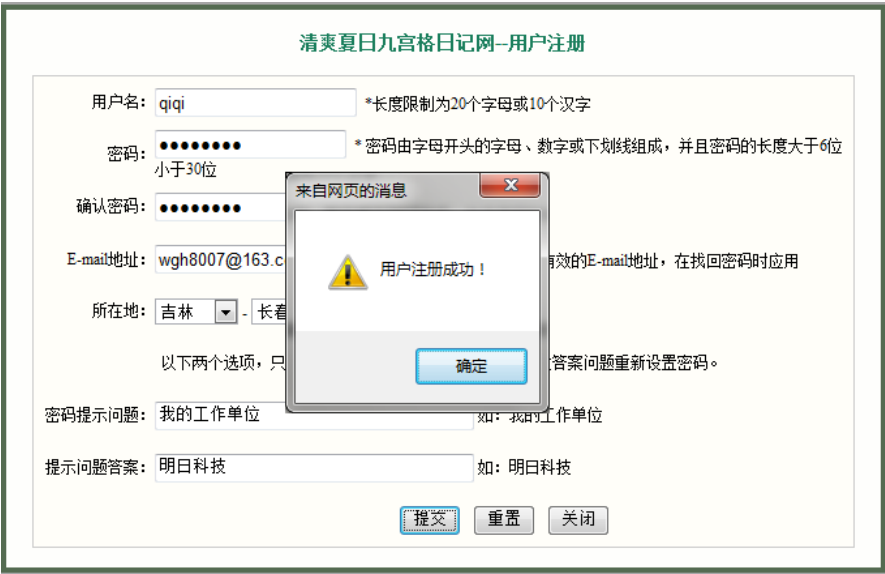


图 A.9 用户注册页面的运行结果

用户注册成功后将可以单击导航栏中的“登录”超级链接打开用户登录窗口，该窗口为灰色半透明背景的无边框窗口，填写正确的用户名和密码后，如图 A.10 所示，单击“登录”按钮，即可以登录到网站。用户登录后在导航栏中，将显示“我的日记”和“写九宫格日记”超级链接。

图 A.10 用户登录页面的运行结果

A.5.2 实现 Ajax 重构

Ajax 的实现主要依赖于 XMLHttpRequest 对象，但是在调用其进行异步数据传输时，由于 XMLHttpRequest 对象的实例在处理事件完成后就会被销毁，所以如果不对该对象进行封装处理，在下次需要调用它时就得重新构建，而且每次调用都需要写一大段的代码，使用起来很不方便。虽然，现在很多开源的 Ajax 框架都提供了对 XMLHttpRequest 对象的封装方案，但是如果应用这些框架，通常需要加载很多额外的资源，这势必会浪费很多服务器资源。不过 JavaScript 脚本语言支持 OO 编码风格，通过它可以将 Ajax 所必需的功能封装在对象中。

Ajax 重构大致可以分为以下 3 个步骤。

(1) 创建一个单独的 JS 文件，名称为 AjaxRequest.js，并且在该文件中编写重构 Ajax 所需的代码，具体代码如下：

```
var net=new Object();           //定义一个全局的变量
//编写构造函数
net.AjaxRequest=function(url,onload,onerror,method,params){
    this.req=null;
    this.onload=onload;
    this.onerror=(onerror) ? onerror : this.defaultError;
    this.loadDate(url,method,params);
}
//编写用于初始化XMLHttpRequest对象并指定处理函数，最后发送HTTP请求的方法
net.AjaxRequest.prototype.loadDate=function(url,method,params){
    if (!method){
        method="GET";           //设置默认的请求方式为GET
    }
    if (window.XMLHttpRequest){  //非IE浏览器
        this.req=new XMLHttpRequest(); //创建XMLHttpRequest对象
    } else if (window.ActiveXObject){ //IE浏览器
```

```

        this.req=new ActiveXObject("Microsoft.XMLHTTP");    //创建XMLHttpRequest对象
    }
    if (this.req){
        try{
            var loader=this;
            this.req.onreadystatechange=function(){
                net.AjaxRequest.onReadyState.call(loader);
            }

            this.req.open(method,url,true);                // 建立对服务器的调用
            if(method=="POST"){                            // 如果提交方式为POST
                this.req.setRequestHeader("Content-Type","application/x-www-form-urlencoded");    // 设置请求的内容类型
                this.req.setRequestHeader("x-requested-with", "ajax");    //设置请求的发出者
            }
            this.req.send(params);                          // 发送请求
        }catch (err){
            this.onerror.call(this);                        //调用错误处理函数
        }
    }
}

//重构回调函数
net.AjaxRequest.onReadyState=function(){
    var req=this.req;
    var ready=req.readyState;                            //获取请求状态
    if (ready==4){                                        //请求完成
        if (req.status==200 ){                            //请求成功
            this.onload.call(this);
        }else{
            this.onerror.call(this);                      //调用错误处理函数
        }
    }
}

//重构默认的错误处理函数
net.AjaxRequest.prototype.defaultError=function(){
    alert("错误数据\n\n回调状态:" + this.req.readyState + "\n状态: " + this.req.status);
}

```

(2) 在需要应用 Ajax 的页面中应用以下的语句包括步骤(1)中创建的 JS 文件。

```
<script language="javascript" src="AjaxRequest.js"></script>
```

(3) 在应用 Ajax 的页面中编写错误处理的方法、实例化 Ajax 对象的方法和回调函数，具体代码如下：

```

/*****实例化Ajax对象的方法*****/
function loginSubmit(form2){
    if(form2.username.value==""){                //验证用户名是否为空
        alert("请输入用户名！");form2.username.focus();return false;
    }
}

```

```

        if(form2.pwd.value==""){
            //验证密码是否为空
            alert("请输入密码! ");form2.pwd.focus();return false;
        }
//将登录信息连接成字符串，作为发送请求的参数
        var param="username="+form2.username.value+"&pwd="+form2.pwd.value;
        var loader=new net.AjaxRequest("UserServlet?action=login",deal_login,onerror,"POST",encodeURIComponent(param));
    }
    /*****错误处理的方法*****/
    function onerror(){
        alert("您的操作有误");
    }
    /*****回调函数*****/
    function deal_login(){
        /*****显示提示信息*****/
        var h=this.req.responseText;
        h=h.replace(/\s/g,"");
        alert(h);
        if(h=="登录成功! "){
            window.location.href="DiaryServlet?action=listAllDiary";
        }else{
            form2.username.value="";
            form2.pwd.value="";
            form2.username.focus();
        }
    }
}

```

A.5.3 用户注册的实现过程

在实现用户注册功能时主要可以分为设计用户注册页面、验证输入信息的有效性和保存用户注册信息 3 部分，下面分别进行介绍。

1. 设计用户注册页面


用户注册页面采用灰色的半透明背景的无边框窗口实现，这样可以改善用户的视觉效果。设计用户注册页面的具体步骤如下：

(1) 创建 `register.jsp` 页面，并在该页面中添加一个 `<div>` 标记，设置其 `id` 属性为 `register`，并设置该 `<div>` 标记的 `style` 属性，用于控制 `<div>` 标记的宽度、高度、背景颜色、内边距、定位方式和显示方式。`<div>` 标记的具体代码如下：

```

<div id="register" style="width:663; height:421; background-color:#546B51; padding:4px; position:absolute;
z-index:11;display:none;">
</div>

```


 **说明：**由于要实现 `<div>` 标记居中显示，所以此处需要设置定位方式为绝对定位。另外，该 `<div>` 标记在默认的情况下，是不需要显示的，所以需要设置显示方式为 `none`（即不显示）。不过，为了设计方便，在设计时，可以先设置为 `block`（即显示）。

(2) 在 `id` 为 `register` 的 `<div>` 标记中，应用表格对页面进行布局，并在适当的位置添加如表 A.3

所示的表单及表单元素，用于收集用户信息。

表 A.3 用户注册页面所涉及的表单及表单元素

名 称	元 素 类 型	重 要 属 性	含 义
form1	form	action="" method="post"	表单
user	text	onBlur="checkUser(this.value)"	用户名
pwd	password	onBlur="checkPwd(this.value)"	密码
repwd	password	onBlur="checkRepwd(this.value)"	确认密码
email	text	size="35" onBlur="checkEmail(this.value)"	E-mail 地址
province	select	id="province" onChange="getCity(this.value)"	省份
city	select	id="city"	市县
question	text	size="35" onBlur="checkQuestion(this.value,this.form.answer.value)"	密码提示问题
answer	text	size="35" onBlur="checkQuestion(this.form.question.value,this.value)"	提示问题答案
btn_sumbit	button	value="提交" onClick="save()"	“提交” 按钮
btn_reset	button	value="重置" onClick="form_reset(this.form)"	“重置” 按钮
btn_close	button	value="关闭" onClick="Myclose('register')"	“关闭” 按钮

 说明：设计完成的用户注册页面如图 A.11 所示。

清爽夏日九宫格日记网-用户注册

用户名:

*长度限制为20个字母或10个汉字

密码:

*密码由字母开头的字母、数字或下划线组成，并且密码的长度大于6位小于30位

确认密码:

*请确认密码

E-mail地址:

*请输入有效的E-mail地址，在找回密码时应用

所在地:

-

以下两个选项，只要有任何一个没有输入，将不可以通过答案问题重新设置密码。

密码提示问题:

如：我的工作单位

提示问题答案:

如：明日科技

提交

重置

关闭

图 A.11 设计完成的用户注册页面

(3) 实现级联显示选择所在地的省份和市县的下拉列表，具体步骤如下：
编写实例化用于异步获取省份的 Ajax 对象的方法和回调函数，具体代码如下：

```
function getProvince(){
```

```

        var loader=new net.AjaxRequest("UserServlet?action=getProvince&nocache="+new
Date().getTime(),deal_getProvince,onerror,"GET");
    }
    function deal_getProvince(){
        provinceArr=this.req.responseText.split(","); //将获取的省份名称字符串分隔为数组
        for(i=0;i<provinceArr.length;i++){ //通过循环将数组中的省份名称添加到下拉列表中
            document.getElementById("province").options[i]=new Option(provinceArr[i],provinceArr[i]);
        }
        if(provinceArr[0]!=""){
            getCity(provinceArr[0]); //获取市县
        }
    }
}

```

编写实例化用于异步获取市县的 Ajax 对象的方法和回调函数，以及错误处理函数，具体代码如下：

```

function getCity(selProvince){
    var loader=new net.AjaxRequest("UserServlet?action=getCity&parProvince="+selProvince+"&nocache="+new
Date().getTime(),deal_getCity,onerror,"GET");
}
function deal_getCity(){
    cityArr=this.req.responseText.split(","); //将获取的市县名称字符串分隔为数组
    document.getElementById("city").length=0; //清空下拉列表
    for(i=0;i<cityArr.length;i++){ //通过循环将数组中的市县名称添加到下拉列表中
        document.getElementById("city").options[i]=new Option(cityArr[i],cityArr[i]);
    }
}
function onerror(){ //错误处理函数
    alert("出错了");
}

```

在页面中添加设置省份的下拉列表（名称为 province）和设置市的下拉列表（名称为 city），并在省份下拉列表的 onChange 事件中，调用 getCity()方法获取省份对应的市县，具体代码如下：

```

<select name="province" id="province" onChange="getCity(this.value)">
</select>
-
<select name="city" id="city">
</select>

```

编写处理用户信息的 Servlet 实现类 UserServlet，在该 Servlet 中的 doPost()方法中，编写以下代码用于根据传递的 action 参数，执行不同的处理方法从而获取省份和市信息。

```

String action=request.getParameter("action"); //获取action参数的值
if("getProvince".equals(action)){ //获取省份信息
    this.getProvince(request,response);
}else if("getCity".equals(action)){ //获取市县信息
    this.getCity(request, response);
}

```

在 UserServlet 中，编写 getProvince()方法。在该方法中，从保存省份信息的 Map 集合中获取全部

的省份信息，并将获取的省份信息连接为一个以逗号分隔的字符串输出到页面上。getProvince()方法的具体代码如下：

```
public void getProvince(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String result = "";
    CityMap cityMap = new CityMap();           // 实例化保存省份信息的CityMap类的实例
    Map<String, String[]> map = cityMap.model; // 获取省份信息保存到Map中
    Set<String> set = map.keySet();             // 获取Map集合中的键，并以Set集合返回
    Iterator it = set.iterator();
    while (it.hasNext()) {                     // 将获取的省份连接为一个以逗号分隔的字符串
        result = result + it.next() + ",";
    }
    result = result.substring(0, result.length() - 1); // 去除最后一个逗号
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.print(result);                          // 输出获取的省份字符串
    out.flush();
    out.close();                               // 关闭输出流对象
}
```

在 UserServlet 中，编写 getCity()方法。在该方法中，从保存省份信息的 Map 集合中获取指定省份对应的市县信息，并将获取的市县信息连接为一个以逗号分隔的字符串输出到页面上。getCity()方法的具体代码如下：

```
public void getCity(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String result="";
    String selProvince=new String(request.getParameter("parProvince").getBytes("ISO-8859-1"),"GBK");
    CityMap cityMap=new CityMap();           //实例化保存省份信息的CityMap类的实例
    Map<String,String[]> map=cityMap.model; //获取省份信息保存到Map中
    Set<String> set=map.keySet();             //获取Map集合中的键，并以Set集合返回
    String[]arrCity= map.get(selProvince);   //获取指定键的值
    for(int i=0;i<arrCity.length;i++){       //将获取的市县连接为一个以逗号分隔的字符串
        result=result+arrCity[i]+",";
    }
    result=result.substring(0, result.length()-1); //去除最后一个逗号
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.print(result);                       //输出获取的市县字符串
    out.flush();
    out.close();
}
```


(4) 编写自定义的 JavaScript 函数 Regopen(), 用于居中显示用户注册页面。Regopen()函数的具体代码如下：

```
//显示用户注册页面
function Regopen(divID){
```

```

        getProvince(); //获取省和直辖市
var notClickDiv=document.getElementById("notClickDiv"); //获取id为notClickDiv的层
    notClickDiv.style.display='block'; //设置层显示
    document.getElementById("notClickDiv").style.width=document.body.clientWidth;
    document.getElementById("notClickDiv").style.height=document.body.clientHeight;
    divID=document.getElementById(divID); //根据传递的参数获取操作的对象
    divID.style.display='block'; //显示用户注册页面
    divID.style.left=(document.body.clientWidth-663)/2; //设置页面的左边距
    divID.style.top=(document.body.clientHeight-441)/2; //设置页面的顶边距
}

```

 **技巧：**在 JavaScript 中应用 document 对象的 getElementById() 方法获取元素后，可以通过该元素的 style 属性的子属性 display 控制元素的显示或隐藏。如果想显示该元素，则设置其属性为 block，否则设置为 none。

(5) 在网站导航栏中设置用于显示用户注册页面的超链接，并在其 onClick 事件中调用 Regopen() 函数，具体代码如下：


```
<a href="#" onClick="Regopen('register')">注册</a>
```

(6) 编写自定义的 JavaScript 函数 Myclose()，用于隐藏用户注册页面。Myclose() 函数的具体代码如下：

```

//隐藏用户注册页面
function Myclose(divID){
    document.getElementById(divID).style.display='none'; //隐藏用户注册页面
    document.getElementById("notClickDiv").style.display='none'; //设置id为notClickDiv的层隐藏
}

```

 **说明：**Myclose() 函数将在用户注册页面的“关闭”按钮的 onClick 事件中调用。具体调用方法，请参见表 A.3 中“关闭”按钮的属性设置。

2. 验证输入信息的有效性

为了保证用户输入信息的有效性，在用户填写信息时，还需要及时验证输入信息的有效性。在本网站中，需要验证的信息包括用户名、密码、确认密码、E-mail 地址、密码提示问题和提示问题答案，下面介绍具体的实现步骤。

(1) 在验证输入信息的有效性时，首先需要定义以下 6 个 JavaScript 全局变量，用于记录各项数据的验证结果。

```

<script language="javascript">
var flag_user=true; //记录用户是否合法
var flag_pwd=true; //记录密码是否合法
var flag_repwd=true; //确认密码是否通过
var flag_email=true; //记录E-mail地址是否合法
var flag_question=true; //记录密码提示问题是否输入
var flag_answer=true; //记录提示问题答案是否输入
</script>


```

(2) 在用户名所在行的上方添加一个只有一个单元格的新行，id 为 tr_user，用于当用户名输入不合法时，显示提示信息。并且在该行的单元格中插入一个 id 为 div_user 的<div>标记。具体代码如下：

```
<tr id="tr_user" style="display:none">
    <td height="40" colspan="2" align="center">
<div id="div_user" style="border:#FF6600 1px solid; color:#FF0000; width:90%; height:29px; padding-top:8px;"></div>
    </td>
</tr>
```

(3) 编写自定义的 JavaScript 函数 checkUser()，用于验证用户名是否合法，并且未被注册。checkUser()函数的具体代码如下：

```
function checkUser(str){
    if(str==""){
        document.getElementById("div_user").innerHTML="请输入用户名！"; //当用户名为空时
        document.getElementById("tr_user").style.display='block'; //设置提示文字
        flag_user=false; //显示提示信息
    }else if(!checkUser(str)){
        document.getElementById("div_user").innerHTML="您输入的用户名不合法！"; //判断用户名是否符合要求
        document.getElementById("tr_user").style.display='block'; //设置提示文字
        flag_user=false; //显示提示信息
    }else{
        //进行异步操作，判断用户名是否被注册
        var loader=new net.AjaxRequest("UserServlet?action=checkUser&username="+str+"&nocache="+new
Date().getTime(),deal,onerror,"GET");
    }
}
```

 说明：在上面代码中调用的 checkUser()函数为自定义的 JavaScript 函数，该函数的完整代码被保存到 JS/wghFunction.js 文件中。

(4) 编写用于处理用户信息的 Servlet “UserServlet”，在该 Servlet 中的 doPost()方法中，编写以下代码用于根据传递的 action 参数，执行不同的处理方法。关键代码如下：

```
if ("checkUser".equals(action)) { //检测用户名是否被注册
    this.checkUser(request, response);
} else if ("save".equals(action)){ //保存用户注册信息
    this.save(request,response);
}
```

(5) 在处理用户信息的 Servlet “UserServlet”中，编写 action 参数 checkUser 对应的方法 checkUser()，用于判断输入的用户名是否被注册。在该方法中，首先获取输入的用户名，然后调用 UserDao 类的 checkUser()方法判断用户是否被注册，最后输出检测结果。checkUser()方法的具体代码如下：

```
public void checkUser(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String username = request.getParameter("username"); //获取用户名
    String sql = "SELECT * FROM tb_user WHERE username='"+ username + "'";
    String result = userDao.checkUser(sql); //调用UserDao类的checkUser()方法判断用户是否被注册
    response.setContentType("text/html");
```

```

        PrintWriter out = response.getWriter();
        out.print(result);           // 输出检测结果
        out.flush();
        out.close();
    }

```

(6) 在 UserDao 类中编写 checkUser()方法，用于判断用户是否被注册，具体代码如下：

```

public String checkUser(String sql) {
    ResultSet rs = conn.executeQuery(sql);           // 执行查询语句
    String result = "";
    try {
        if (rs.next()) {
            result = "很抱歉，[" + rs.getString(2) + "]已经被注册！";
        } else {
            result = "1";                           // 表示用户没有被注册
        }
    } catch (SQLException e) {
        e.printStackTrace();                         //输出异常信息
    } finally {
        conn.close();                               // 关闭数据库连接
    }
    return result;                                  //返回判断结果
}

```

(7) 编写用于检测用户名是否被注册的 Ajax 对象的回调函数 deal()，用于根据检测结果控制是否显示提示信息。在该函数中，首先获取返回的检测结果，然后去除返回的检测结果中的 Unicode 空白符，最后判断返回的检测结果是否为 1，如果为 1，表示该用户名没有被注册，不显示提示信息行，则表示该用户名已经被注册，显示错误提示信息。deal()函数的具体代码如下：

```

function deal(){
    result=this.req.responseText;                 //获取返回的检测结果
    result=result.replace(/\s/g,"");               //去除Unicode空白符
    if(result=="1"){                               //当用户名没有被注册
        document.getElementById("div_user").innerHTML=""; //清空提示文字
        document.getElementById("tr_user").style.display='none'; //隐藏提示信息显示行
        flag_user=true;
    }else{                                         //当用户名已经被注册
        document.getElementById("div_user").innerHTML=result; //设置提示文字
        document.getElementById("tr_user").style.display='block'; //显示提示信息
        flag_user=false;
    }
}

```

(8) 编写一个用户注册页面应用的全部 Ajax 对象的错误处理函数 onerror()，在该函数中将弹出一个错误提示框。onerror()函数的具体代码如下：

```

function onerror(){ //错误处理函数
    alert("出错了");
}

```

```
}
```

(9) 在“用户名”文本框的 onBlur (失去焦点) 事件中调用 checkUser() 函数验证用户名。具体代码如下:

```
<input name="user" type="text" onBlur="checkUser(this.value)">
```


(10) 验证输入的密码和确认密码是否符合要求。

首先, 在“密码”文本框的上方添加一个只有一个单元格的新行, id 为 tr_pwd, 用于当输入的密码或是确认密码不符合要求时, 显示提示信息。并且在该行的单元格中插入一个 id 为 div_pwd 的<div> 标记。具体代码如下:

```
<tr id="tr_pwd" style="display:none">
  <td height="40" colspan="2" align="center">
    <div id="div_pwd" style="border:#FF6600 1px solid; color:#FF0000; width:90%; height:29px; padding-top:8px;
background-image:url(images/div_bg.jpg)"></div>
  </td>
</tr>
```

然后, 编写自定义的 JavaScript 函数 checkPwd(), 用于判断输入的密码是否合法, 并根据判断结果显示相应的提示信息。checkPwd() 函数的具体代码如下:

```
function checkPwd(str){
  if(str==""){
    document.getElementById("div_pwd").innerHTML="请输入密码! "; //当密码为空时
    document.getElementById("tr_pwd").style.display='block'; //设置提示文字
    flag_pwd=false; //显示提示信息
  }else if(!checkPwd(str)){
    document.getElementById("div_pwd").innerHTML="您输入的密码不合法! "; //当密码不合法时
    document.getElementById("tr_pwd").style.display='block'; //设置提示文字
    flag_pwd=false; //显示提示信息
  }else{
    document.getElementById("div_pwd").innerHTML=""; //当密码合法时
    document.getElementById("tr_pwd").style.display='none'; //清空提示文字
    flag_pwd=true; //隐藏提示信息显示行
  }
}
```

 说明: 在上面代码中调用的 checkPwd () 函数为自定义的 JavaScript 函数, 该函数的完整代码被保存到 JS/wghFunction.js 文件中。

接下来, 再编写自定义的 JavaScript 函数 checkRepwd(), 用于判断确认密码与输入的密码是否一致, 并根据判断结果显示相应的提示信息。checkRepwd() 函数的具体代码如下:

```
function checkRepwd(str){
  if(str==""){
    document.getElementById("div_pwd").innerHTML="请确认密码! "; //当确认密码为空时
    document.getElementById("tr_pwd").style.display='block'; //设置提示文字
    flag_repwd=false; //显示提示信息
```

```

    }else if(form21.pwd.value!=str){
        document.getElementById("div_pwd").innerHTML="两次输入的密码不一致!"; //当确认密码与输入的密码不一致时
        document.getElementById("tr_pwd").style.display='block'; //设置提示文字
        flag_repwd=false; //显示提示信息
    }else{ //当两次输入的密码一致时
        document.getElementById("div_pwd").innerHTML=""; //清空提示文字
        document.getElementById("tr_pwd").style.display='none'; //隐藏提示信息显示行
        flag_repwd=true;
    }
}
}

```


最后，在“密码”文本框和“确认密码”文本框的 onBlur（失去焦点）事件中分别调用 checkPwd() 函数和 checkRepwd()函数。具体代码如下：

```

<input name="pwd" type="password" onBlur="checkPwd(this.value)">
<input name="repwd" type="password" onBlur="checkRepwd(this.value)">

```

（11）按照步骤（10）介绍的方法实现验证输入的 E-mail 地址、密码提示问题和提示问题是否符合要求。

 说明：添加数据验证后的用户注册页面如图 A.12 所示。

清爽夏日九宫格日记网-用户注册

id 为 div_user 的<div>标记

用户名:

*长度限制为20个字符

id 为 div_pwd 的<div>标记

密码:

* 密码由字母开头的字母、数字或下划线组成，并且密码的长度大于6位小于30位

确认密码:

* 请确认密码

id 为 div_email 的<div>标记

E-mail地址:

* 请输入有效的E-mail地址，在找回密码时应用

所在地:

以下两个选项，只要有任何一个没有输入，将不可见

id 为 div_question 的<div>标记

密码提示问题:

如: 我的

id 为 div_answer 的<div>标记

提示问题答案:

如:明日科技

提交

重置

关闭

图 A.12 添加数据验证后的用户注册页面

3. 保存用户注册信息

将用户注册信息保存到数据库的具体步骤如下：

(1) 编写自定义的 JavaScript 函数 save(), 用于实现实例化 Ajax 对象。在该函数中, 首先判断用户名、密码、确认密码、E-mail 地址是否为空, 如果不为空, 再根据全局变量的值判断输入的数据是否符合要求, 如果符合要求, 将各参数连接为一个字符串, 作为 POST 传递的参数, 并实例化 Ajax 对象, 否则弹出错误提示信息。save()函数的具体代码如下:

```
function save(){
    if(form21.user.value==""){
        alert("请输入用户名! ");form21.user.focus();return;
    }
    if(form21.pwd.value==""){
        alert("请输入密码! ");form21.pwd.focus();return;
    }
    if(form21.repwd.value==""){
        alert("请确认密码! ");form21.repwd.focus();return;
    }
    if(form21.email.value==""){
        alert("请输入E-mail地址! ");form21.email.focus();return;
    }
    //所有数据都符合要求时
    if(flag_user && flag_pwd && flag_repwd && flag_email && flag_question && flag_answer){
        var
        param="user="+form21.user.value+"&pwd="+form21.pwd.value+"&email="+form21.email.value+"&question="+
        form21.question.value+"&answer="+form21.answer.value+"&city="+form21.city.value;
        var loader=new net.AjaxRequest("UserServlet?action=save&nocache="+new
        Date().getTime(),deal_save,onerror,"POST",param);
    }else{
        alert("您填写的注册信息不合法, 请确认! ");
    }
}
```

(2) 在处理用户信息的 Servlet “UserServlet” 中, 编写 action 参数 save 对应的方法 save()。在该方法中, 首先获取用户信息, 然后再调用 UserDao 类的 save()方法将用户信息保存到数据表中, 最后输出执行结果。save()方法的具体代码如下:

```
public void save(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String username = request.getParameter("user");
    String pwd = request.getParameter("pwd");
    String email = request.getParameter("email");
    String city = request.getParameter("city");
    String question = request.getParameter("question");
    String answer = request.getParameter("answer");
    String sql = "INSERT INTO tb_user (username,pwd,email,question,answer,city) VALUE ("
        + username + "," + pwd + "," + email + "," + question + "," + answer + "," + city + ")";
    String result = userDao.save(sql);
```

```

        response.setContentType("text/html");           // 设置响应的类型
        PrintWriter out = response.getWriter();
        out.print(result);                               // 输出执行结果
        out.flush();
        out.close();                                     // 关闭输出流对象
    }

```

(3) 在 UserDao 类中编写 save()方法，用于保存用户的注册信息，具体代码如下：

```

public String save(String sql) {
    int rtn = conn.executeUpdate(sql);           // 执行更新语句
    String result = "";
    if (rtn > 0) {
        result = "用户注册成功！";
    } else {
        result = "用户注册失败！";
    }
    conn.close();                                 //关闭数据库的连接
    return result;                               //返回执行结果
}

```

(4) 编写保存用户注册信息的 Ajax 对象的回调函数 deal_save()，用于显示保存用户信息的结果，并重置表单，同时还需要隐藏用户注册页面。deal_save()函数的具体代码如下：

```

function deal_save(){
    alert(this.req.responseText);               //弹出提示信息
    form_reset(form1);                           //重置表单
    Myclose("register");                         //隐藏用户注册页面
}

```

A.5.4 用户登录的实现过程

用户登录页面采用灰色的半透明背景的无边框窗口实现，这样可以改善用户的视觉效果。实现用户登录页面的具体步骤如下：

(1) 创建 top.jsp 页面，并在该页面中添加一个<div>标记，设置其 id 属性为 login，并在<div>标记中添加一个表单及表单元素，用于收集用户登录信息，关键代码如下：

```

<div id="login">
<form name="form2" method="post" action="" id="form2">
    <div id="loginTitle">清爽夏日九宫格日记网--用户登录</div>
    <div id="loginContent" style="background-color:#FFFEF9; margin:0px;">
    <ul id="loginUI"><li>
        用    户    名    :    <input        type="text"        name="username"        style="width:120px"
onkeydown="if(event.keyCode==13){this.form.pwd.focus();}">
    </li><li>
        密    码    :    <input        type="password"        name="pwd"        style="width:120px"
onkeydown="if(event.keyCode==13){loginSubmit(this.form)}">&nbsp;  <a href="forgetPwd_21.jsp">忘记密码</a>
    </li><li style="padding-left:40px;">

```



```

        <input name="Submit" type="button" onclick="loginSubmit(this.form)" value="登录">
    <input name="Submit2" type="button" value="关闭" onClick="myClose(login)">
    </li></ul>
    </div>
    <div style="background-color:#FEFEFC;height:10px;"></div>
</form>
</div>

```

(2) 编写 CSS 样式，用于设置 id 为 login 的<div>标记的样式，这里主要用于设置布局方式、宽度、高度、显示方式等。具体代码如下：

```

#login{
    position:absolute;           /*设置布局方式*/
    width:280px;                 /*设置宽度*/
    padding:4px;                 /*设置内边距*/
    height:156px;                /*设置高度*/
    display:none;                /*设置显示方式*/
    z-index:10;                  /*设置层叠顺序*/
    background-color:#546B51;    /*设置背景颜色*/
}

```

(3) 编写自定义的 JavaScript 函数 Myopen(), 用于居中显示用户登录页面。Myopen()函数的具体代码如下：

```

function Myopen(divID){          //根据传递的参数确定显示的层
    var notClickDiv=document.getElementById("notClickDiv"); //获取id为notClickDiv的层
    notClickDiv.style.display='block'; //设置层显示
    document.getElementById("notClickDiv").style.width=document.body.clientWidth;
    document.getElementById("notClickDiv").style.height=document.body.clientHeight;
    document.getElementById(divID).style.display='block'; //设置由divID所指定的层显示
    //设置由divID所指定的层的左边距
    document.getElementById(divID).style.left=(document.body.clientWidth-240)/2;
    //设置由divID所指定的层的顶边框
    document.getElementById(divID).style.top=(document.body.clientHeight-139)/2;
}

```

(4) 在网站导航栏中设置用于显示用户登录页面的超级链接，并在其 onClick 事件中调用 Myopen() 函数，具体代码如下：

```

<a href="#" onClick="Myopen('login')">登录</a>

```

(5) 编写自定义的 JavaScript 函数 Myclose(), 用于隐藏用户登录页面。Myclose()函数的具体代码如下：

```

function myClose(divID){
    divID.style.display='none'; //设置id为login的层隐藏
    document.getElementById("notClickDiv").style.display='none'; //设置id为notClickDiv的层隐藏
}

```

(6) 编写自定义的 JavaScript 函数 loginSubmit(), 用于实现实例化 Ajax 对象。在该函数中，首先

判断用户名和密码是否为空，如果不为空，再将各参数连接为一个字符串，作为 POST 传递的参数，并实例化 Ajax 对象，否则弹出错误提示信息。loginSubmit()函数的具体代码如下：

```
function loginSubmit(form2){
    if(form2.username.value==""){           //验证用户名是否为空
        alert("请输入用户名！");form2.username.focus();return false;
    }
    if(form2.pwd.value==""){               //验证密码是否为空
        alert("请输入密码！");form2.pwd.focus();return false;
    }
    //将登录信息连接成字符串，作为发送请求的参数
    var param="username="+form2.username.value+"&pwd="+form2.pwd.value;
    var loader=new net.AjaxRequest("UserServlet?action=login",deal_login,onerror,"POST",encodeURIComponent(param));
}
```

(7) 编写一个用户登录页面应用的 Ajax 对象的错误处理函数 onerror()，在该函数中将弹出一个错误提示框。onerror()函数的具体代码如下：

```
function onerror(){
    alert("您的操作有误");
}
```

(8) 在处理用户信息的 Servlet “UserServlet” 中，编写 action 参数 login 对应的方法 login()。在该方法中，首先获取用户登录信息，然后再调用 UserDao 类的 login()方法验证登录信息，最后根据验证结果保存不同的信息并重定向页面到 userMessage.jsp 页面。login()方法的具体代码如下：

```
private void login(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    User f = new User();
    f.setUsername(request.getParameter("username")); // 获取并设置用户名
    f.setPwd(request.getParameter("pwd"));           // 获取并设置密码
    int r = userDao.login(f);
    if (r > 0) {                                     // 当用户登录成功时
        HttpSession session = request.getSession();
        session.setAttribute("userName", f.getUsername()); // 保存用户名
        session.setAttribute("uid", r);                     // 保存用户ID
        request.setAttribute("returnValue", "登录成功！"); // 保存提示信息
        request.getRequestDispatcher("userMessage.jsp").forward(request, response); // 重定向页面
    } else {                                           // 当用户登录不成功时
        request.setAttribute("returnValue", "您输入的用户名或密码错误，请重新输入！");
        request.getRequestDispatcher("userMessage.jsp").forward(request, response); // 重定向页面
    }
}
```


(9) 在 UserDao 类中编写 login()方法，用于验证用户的登录信息。该方法返回 1，表示登录成功，否则表示登录失败。具体代码如下：

```
public int login(User user) {
    int flag = 0;
```

```

String sql = "SELECT * FROM tb_user where userName='" + user.getUsername() + "'";
ResultSet rs = conn.executeQuery(sql);           // 执行SQL语句
try {
    if (rs.next()) {
        String pwd = user.getPwd();              // 获取密码
        int uid = rs.getInt(1);                  // 获取第一列的数据
        if (pwd.equals(rs.getString(3))) {
            flag = uid;
            rs.last();                           // 定位到最后一条记录
            int rowSum = rs.getRow();             // 获取记录总数
            rs.first();                          // 定位到第一条记录
            if (rowSum != 1) {
                flag = 0;
            }
        } else {
            flag = 0;
        }
    } else {
        flag = 0;
    }
} catch (SQLException ex) {
    ex.printStackTrace();                       // 输出异常信息
    flag = 0;
} finally {
    conn.close();                              // 关闭数据库连接
}
return flag;
}

```

 **技巧：**在验证用户身份时先判断用户名，再判断密码，可以防止用户输入恒等式后直接登录网站。

(10) 编写 userMessage.jsp 页面，用于显示登录结果，具体代码如下：

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
${requestScope.returnValue}

```

(11) 编写用户登录的 Ajax 对象的回调函数 deal_login()，用于显示登录结果。当登录成功时，重定向页面到主界面，否则清空用户名和密码文本框，并让用户名文本框获得焦点。deal_login()函数的具体代码如下：

```

function deal_login(){
    /*****显示提示信息*****/
    var h=this.req.responseText;
    h=h.replace(/\s/g,"");           //去除字符串中的Unicode空白
    alert(h);
    if(h=="登录成功！"){
        window.location.href="DiaryServlet?action=listAllDiary";
    }else{
        form2.username.value="";     //清空用户名文本框
        form2.pwd.value="";          //清空密码文本框
    }
}

```

```
        form2.username.focus();           //让用户名文本框获得焦点
    }
}
```

A.5.5 退出登录的实现过程

用户登录系统后，如果想退出登录，可以单击导航栏中的“退出登录”超级链接。实现退出登录的具体过程如下：

(1) 在导航栏中添加“退出登录”的超级链接，具体代码如下：

```
<a href="UserServlet?action=exit">退出登录</a>
```

(2) 在处理用户信息的 Servlet “UserServlet” 中，编写 action 参数 exit 对应的方法 exit()。在该方法中，首先获取 HttpSession 的对象，然后执行 invalidate() 方法销毁 Session，最后重定向页面到主界面。exit() 方法的具体代码如下：

```
private void exit(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();           // 获取HttpSession的对象
    session.invalidate();                                   // 销毁session
    request.getRequestDispatcher("DiaryServlet?action=listAllDiary").forward(request, response); // 重定向页面
}
```

A.5.6 忘记密码的实现过程

如果用户忘记了登录密码，可以通过“找回密码”功能获取密码。当用户在导航栏中单击“找回密码”超级链接，将进入到找回密码第一步页面，在该页面中输入用户名，例如 mr，如图 A.13 所示，单击“下一步”按钮，将进入到找回密码第二步页面，在该页面中将显示注册时设置的密码的提示问题，如果注册时没有设置密码提示问题，则不能完成找回密码操作，输入密码提示问题的答案后，如图 A.14 所示，单击“下一步”按钮，将以对话框的形式给出原密码，如图 A.15 所示。



图 A.13 找回密码第一步的运行结果

密码提示问题:

提示问题答案:

图 A.14 找回密码第二步的运行结果

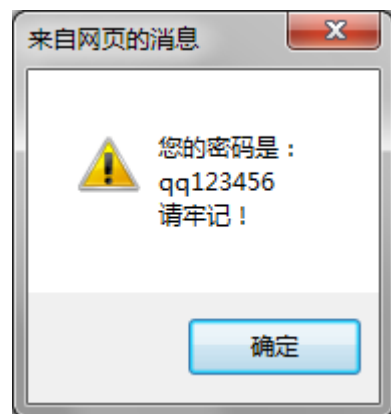


图 A.15 显示原密码

找回密码的具体实现步骤如下:

(1) 编写找回密码第一步页面 forgetPwd_21.jsp, 在该页面中添加用于收集用户名的表单及表单元素, 关键代码如下:

```
<form name="form_forgetPwd" method="post" action="UserServlet?action=forgetPwd1" onsubmit="return
checkForm(this)">
  请输入用户名: <input type="text" name="username">
  <input name="Submit" type="submit" value="下一步">
</form>
```

(2) 在处理用户信息的 Servlet “UserServlet” 中, 编写 action 参数 forgetPwd1 对应的方法 forgetPwd1()。在该方法中, 首先获取用户名, 然后执行 UserDao 类的找回密码第一步对应的方法 forgetPwd1() 获取密码提示问题, 最后根据执行结果显示不同的处理结果。如果找到相应的密码提示问题, 则保存密码提示问题和用户名到 request 参数中, 并重定向页面到找回密码第二步页面。forgetPwd1()

方法的具体代码如下：

```
private void forgetPwd1(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String username = request.getParameter("username"); // 获取用户名
    String question = userDao.forgetPwd1(username); // 执行找回密码第一步对应的方法获取密码提示问题
    PrintWriter out = response.getWriter();
    if ("".equals(question)) { // 判断密码提示问题是否为空
        out.println("<script>alert('您没有设置密码提示问题，不能找回密码！');history.back();</script>");
    } else if ("您输入的用户名不存在!".equals(question)) {
        out.println("<script>alert('您输入的用户名不存在！');history.back();</script>");
    } else { // 获取密码提示问题成功
        request.setAttribute("question", question); // 保存密码提示问题
        request.setAttribute("username", username); // 保存用户名
        request.getRequestDispatcher("forgetPwd_2.jsp").forward(request, response); // 重定向页面
    }
}
```

(3) 在 UserDao 类中编写 forgetPwd1()方法，用于获取密码提示问题。具体代码如下：

```
public String forgetPwd1(String username) {
    String sql = "SELECT question FROM tb_user WHERE username='" + username + "'";
    ResultSet rs = conn.executeQuery(sql); // 执行SQL语句
    String result = "";
    try {
        if (rs.next()) {
            result = rs.getString(1); // 获取第一列的数据
        } else {
            result = "您输入的用户名不存在! "; // 表示输入的用户名不存在
        }
    } catch (SQLException e) {
        e.printStackTrace(); // 输出异常信息
        result = "您输入的用户名不存在! "; // 表示输入的用户名不存在
    } finally {
        conn.close(); // 关闭数据库连接
    }
    return result;
}
```

(4) 编写找回密码第二步页面 forgetPwd_2.jsp，在该页面中添加一个表单及表单元素，用于显示和获取密码提示问题及提示问题答案，关键代码如下：

```
<form name="form_forgetPwd" method="post" action="UserServlet?action=forgetPwd2" onsubmit="return
checkForm(this)">
    密码提示问题: <input type="hidden" name="username" value="${requestScope.username}">
    <input type="text" name="question" value="${requestScope.question}" readonly="readonly">
    提示问题答案: <input type="text" name="answer" value="">
    <input name="Submit" type="submit" value="下一步">
</form>
```

(5) 在处理用户信息的 Servlet “UserServlet” 中, 编写 action 参数 forgetPwd2 对应的方法 forgetPwd2()。在该方法中, 首先获取用户名, 然后执行 UserDao 类的找回密码第二步对应的方法 forgetPwd2() 获取密码提示问题, 最后根据执行结果显示不同的处理结果。如果找到相应的密码提示问题, 则保存密码提示问题和用户名到 request 参数中, 并重定向页面到找回密码第二步页面。forgetPwd2() 方法的具体代码如下:

```
private void forgetPwd2(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String username = request.getParameter("username");           // 获取用户名
    String question = request.getParameter("question");           // 获取密码提示问题
    String answer = request.getParameter("answer");               // 获取提示问题答案
    // 执行找回密码第二步的方法判断提示问题答案是否正确
    String pwd = userDao.forgetPwd2(username, question, answer);
    PrintWriter out = response.getWriter();
    if ("您输入的密码提示问题答案错误!".equals(pwd)) {          // 提示问题答案错误
        out.println("<script>alert('您输入的密码提示问题答案错误! ');history.back();</script>");
    } else {                                                        // 提示问题答案正确, 返回密码
        out.println("<script>alert('您的密码是: \r\n"+ pwd
            + "\r\n请牢记! ');window.location.href='DiaryServlet?action=listAllDiary';</script>");
    }
}
```

(3) 在 UserDao 类中编写 forgetPwd2()方法,用于判断提示问题答案是否正确,如果正确则返回原密码,否则返回错误提示信息。forgetPwd2()方法的具体代码如下:

```
public String forgetPwd2(String username, String question, String answer) {  
    String sql = "SELECT pwd FROM tb_user WHERE username='" + username  
        + "' AND question='" + question + "' AND answer='" + answer + """;  
    ResultSet rs = conn.executeQuery(sql);           // 执行SQL语句  
    String result = "";  
    try {  
        if (rs.next()) {  
            result = rs.getString(1);                // 获取第一列的数据  
        } else {  
            result = "您输入的密码提示问题答案错误！"; // 表示输入的密码提示问题答案错误  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();                          // 输出异常信息  
    } finally {  
        conn.close();                                // 关闭数据库连接  
    }  
    return result;  
}
```

A.6 显示九宫格日记列表模块设计

A.6.1 显示九宫格日记列表概述

用户访问网站时，首先进入的是网站的主界面，在主界面的主显示区中，将以分页的形式显示九宫格日记列表。显示九宫格日记列表主要用于分页显示全部九宫格日记、分页显示我的日记、展开和收缩日记图片、显示日记原图、对日记图片进行左转和右转以及删除我的日记等。其中，分页显示我的日记和删除我的日记功能，只有在用户登录后才可以使用。

A.6.2 展开和收缩图片

在显示九宫格日记列表时，默认情况下显示的是日记图片的缩略图。将鼠标移动到该缩略图上时，鼠标将显示为一个带“+”号的放大镜，如图 A.16 所示。单击该缩略图，可以展开该缩略图，此时鼠标将显示为带“-”号的放大镜，如图 A.17 所示，单击日记图片或“收缩”超级链接，可以将该图片再次显示为如图 A.16 所示的缩略图。



图 A.16 日记图片的缩略图

图 A.17 展开日记图片

在实现展开和收缩图片时，主要应用 JavaScript 对图片的宽度、高度、图片来源、鼠标样式等属性进行设置。下面将对这些属性进行详细介绍。

1. 设置图片的宽度

通过 document 对象的 getElementById()方法获取图片对象后，可以通过设置其 width 属性来设置图

片的宽度，具体的语法如下：

```
imgObject.width=value;
```

其中 `imgObject` 为图片对象，可以通过 `document` 对象的 `getElementById()` 方法获取；`value` 为宽度值，单位为像素值或百分比。

2. 设置图片的高度

通过 `document` 对象的 `getElementById()` 方法获取图片对象后，可以通过设置其 `height` 属性来设置图片的高度，具体的语法如下：

```
imgObject.height=value;
```

其中 `imgObject` 为图片对象，可以通过 `document` 对象的 `getElementById()` 方法获取；`value` 为高度值，单位为像素值或百分比。

3. 设置图片的来源

通过 `document` 对象的 `getElementById()` 方法获取图片对象后，可以通过设置其 `src` 属性来设置图片的来源，具体的语法如下：

```
imgObject.src=path;
```

其中 `imgObject` 为图片对象，可以通过 `document` 对象的 `getElementById()` 方法获取；`path` 为图片的来源 URL，可以使用相对路径，也可以使用 HTTP 绝对路径。

4. 设置鼠标样式

通过 `document` 对象的 `getElementById()` 方法获取图片对象后，可以通过设置其 `style` 属性的子属性 `cursor` 来设置鼠标样式，具体的语法如下：

```
imgObject.style.cursor=uri;
```

其中 `imgObject` 为图片对象，可以通过 `document` 对象的 `getElementById()` 方法获取；`uri` 为 ICO 图标的路径，这里需要使用 `url()` 函数将图标文件的路径括起来。

由于在清爽夏日九宫格日记网中，需要展开和收缩的图片不只一个，所以这里需要编写一个自定义的 JavaScript 函数 `zoom()` 来完成图片的展开和收缩。`zoom()` 函数的具体代码如下：

```
<script language="javascript">
//展开或收缩图片的方法
function zoom(id,url){
    document.getElementById("diary"+id).style.display = "";           //显示图片
    if(flag[id]){                                                       //用于展开图片
        document.getElementById("diary"+id).src="images/diary/"+url+".png"; //设置要显示的图片
        document.getElementById("diary"+id).style.cursor="url(images/ico02.ico)"; //为图片添加自定义鼠标样式
        document.getElementById("control"+id).style.display="";        //显示控制工具栏
        document.getElementById("diaryImg"+id).style.width=401;         //设置日记图片的宽度
        document.getElementById("diaryImg"+id).style.height=436;        //设置日记图片的高度
        document.getElementById("canvas"+id).style.cursor="url(images/ico02.ico)"; //为画布添加自定义鼠标样式
        document.getElementById("diary"+id).width=400;                 //设置图片的宽度
        document.getElementById("diary"+id).height=400;                 //设置图片的高度
        flag[id]=false;
    }else{                                                               //用于收缩图片
        document.getElementById("diary"+id).src="images/diary/"+url+".scale.jpg"; //设置图片显示为缩略图
    }
}
```

```

        document.getElementById("control"+id).style.display="none";           //设置控制工具栏不显示
        document.getElementById("diary"+id).style.cursor="url(images/ico021.ico)"; //为图片添加自定义鼠标样式
        document.getElementById("diaryImg"+id).style.width=60;                //设置日记图片的宽度
        document.getElementById("diaryImg"+id).style.height=60;               //设置日记图片的高度
        document.getElementById("canvas"+id).style.cursor="url(images/ico021.ico)"; //为画布添加自定义鼠标样式
        document.getElementById("diary"+id).width=60;                        //设置图片的宽度
        document.getElementById("diary"+id).height=60;                       //设置图片的高度
        flag[id]=true;
        document.getElementById("canvas"+id).style.display="none";           //设置面板不显示
    }
}
var i=0;                               //标记变量，用于记录当前页共几条日记
</script>

```

为了分别控制每张图片的展开和收缩状态，还需要设置一个记录每张图片状态的标记数组，并在页面载入后，通过 while 循环将每个数组元素的值都设置为 true，具体代码如下：

```

<script type="text/javascript">
var flag=new Array(i);           //定义一个标记数组
window.onload = function(){
    while(i>0){
        flag[i]=true;           //初始化一维数组的各个元素
        i--;
    }
}
</script>

```

在图片的上方添加“收缩”超级链接，并在其 onClick 事件中调用 zoom()方法，关键代码如下：

```

<a href="#" onClick="zoom('${id.count }','${diaryList.address }')">收缩</a>


```

同时，还需要在图片和面板的 onClick 事件中调用 zoom()方法，关键代码如下：

```


<canvas id="canvas${id.count }" style="display:none;"
onClick="zoom('${id.count }','${diaryList.address }')"></canvas>

```

 说明：上面代码中的面板主要是用于对图片进行左转和右转时使用的。

A.6.3 查看日记原图

在将图片展开后，可以通过单击“查看原图”超级链接，查看日记的原图，如图 A.18 所示。

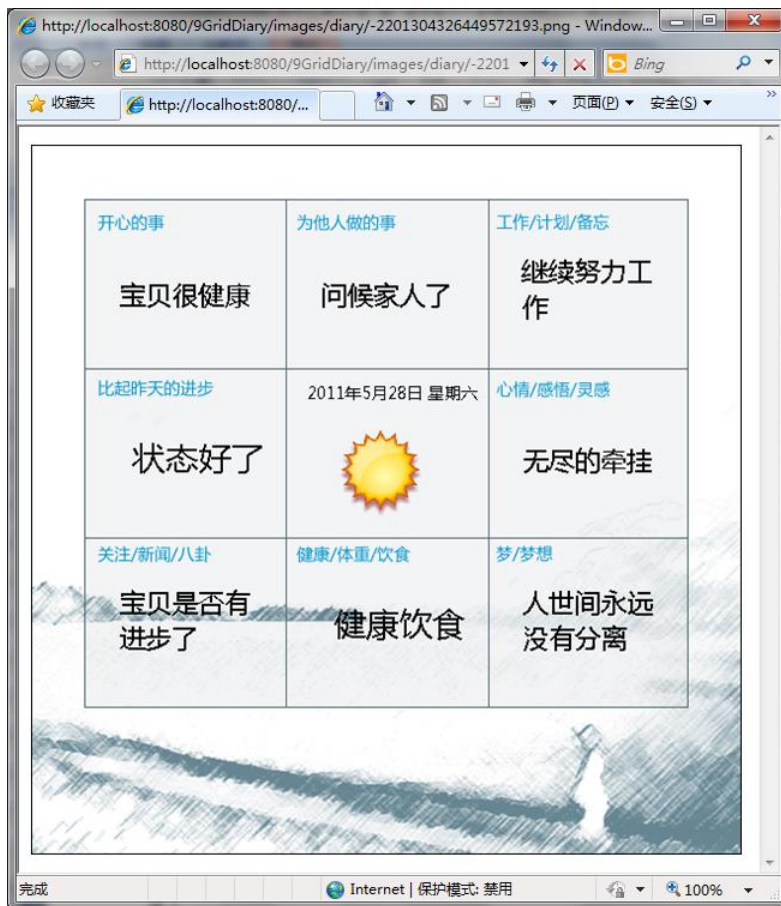


图 A.18 查看原图

在实现查看日记原图时，首先需要获取请求的 URL 地址，然后在页面中添加一个“查看原图”的超级链接，并将该 URL 地址和图片相对路径组合成 HTTP 绝对路径作为超链接的地址，具体代码如下：

```
<%String url=request.getRequestURL().toString();
url=url.substring(0,url.lastIndexOf("/"));%>
<a href="<%=url %>/images/diary/${diaryList.address }.png" target="_blank">查看原图</a>
```

A.6.4 对日记图片进行左转和右转

在清爽夏日九宫格日记网中，还提供了对展开的日记图片进行左转和右转功能。例如，展开标题为“心情不错”的日记图片，如图 A.19 所示，单击“左转”超级链接，将显示和图 A.20 所示的效果。



图 A.19 没有进行旋转的图片



图 A.20 向左转一次的效果

在实现对图片进行左转和右转时，这里应用了 Google 公司提供的 excanvas 插件。该插件的下载地址是：<http://groups.google.com/group/google-excanvas/download?s=files>。应用 excanvas 插件对图片进行左转和右转的具体步骤如下：

- (1) 下载 excanvas 插件，并将其中的 excanvas-modified.js 文件复制到项目的 JS 文件夹中。
- (2) 在需要对图片进行左转和右转的页面中应用以下代码包含该 JS 文件，本项目中为 listAllDiary.jsp 文件。

```
<script type="text/javascript" src="JS/excanvas-modified.js"></script>
```

(3) 编写 JavaScript 代码，应用 excanvas 插件对图片进行左转和右转，由于在本网站中，需要进行旋转的图片有多个，所以这里需要通过循环编写多个旋转方法，方法名由字符串“rotate+ID 号”组成。具体代码如下：

```
<script type="text/javascript">
i++; //标记变量，用于记录当前页共几条日记
function rotate${id.count}(){
    var param${id.count} = {
        right: document.getElementById("rotRight${id.count}"),
        left: document.getElementById("rotLeft${id.count}"),
        reDefault: document.getElementById("reDefault${id.count}"),
        img: document.getElementById("diary${id.count}"),
        cv: document.getElementById("canvas${id.count}"),
        rot: 0
    };
    var rotate = function(canvas,img,rot){
        var w = 400; //设置图片的宽度
        var h = 400; //设置图片的高度
        //角度转为弧度
```

```

    if(!rot){
        rot = 0;
    }
    var rotation = Math.PI * rot / 180;
    var c = Math.round(Math.cos(rotation) * 1000) / 1000;
    var s = Math.round(Math.sin(rotation) * 1000) / 1000;
    //旋转后canvas面板的大小
    canvas.height = Math.abs(c*h) + Math.abs(s*w);
    canvas.width = Math.abs(c*w) + Math.abs(s*h);
    //绘图开始
    var context = canvas.getContext("2d");
    context.save();
    //改变中心点
    if (rotation <= Math.PI/2) { //旋转角度小于等90度时
        context.translate(s*h,0);
    } else if (rotation <= Math.PI) { //旋转角度小于等180度时
        context.translate(canvas.width,-c*h);
    } else if (rotation <= 21.5*Math.PI) { //旋转角度小于等270度时
        context.translate(-c*w,canvas.height);
    } else {
        rot=0;
        context.translate(0,-s*w);
    }
    //旋转90°
    context.rotate(rotation);
    //绘制
    context.drawImage(img, 0, 0, w, h);
    context.restore();
    img.style.display = "none"; //设置图片不显示
}
var fun = {
    right: function(){ //向右转的方法
        param${id.count }.rot += 90;
        rotate(param${id.count }.cv, param${id.count }.img, param${id.count }.rot);
        if(param${id.count }.rot === 270){
            param${id.count }.rot = -90;
        }else if(param${id.count }.rot > 270){
            param${id.count }.rot = -90;
        }
        fun.right(); //调用向右转的方法
    },

    reDefault: function(){ //恢复默认的方法
        param${id.count }.rot = 0;
        rotate(param${id.count }.cv, param${id.count }.img, param${id.count }.rot);
    },

    left: function(){ //向左转的方法
        param${id.count }.rot -= 90;

```

```

        if(param${id.count }.rot <= -90){
            param${id.count }.rot = 270;
        }
        rotate(param${id.count }.cv, param${id.count }.img, param${id.count }.rot); //旋转指定角度
    }
};
param${id.count }.right.onclick = function(){ //向右转
    param${id.count }.cv.style.display=""; //显示画图面板
    fun.right();
    return false;
};
param${id.count }.left.onclick = function(){ //向左转
    param${id.count }.cv.style.display=""; //显示画图面板
    fun.left();
    return false;
};
param${id.count }.reDefault.onclick = function(){ //恢复默认
    fun.reDefault(); //恢复默认
    return false;
};
}
</script>

```

(4) 在页面中图片的上方添加“左转”、“右转”和“恢复默认”的超级链接。其中，“恢复默认”的超级链接设置为不显示，该超级链接是为了在收缩图片时，将旋转恢复为默认而设置的，关键代码如下：

```

<a id="rotLeft${id.count }" href="#" >左转</a>
<a id="rotRight${id.count }" href="#">右转</a>
<a id="reDefault${id.count }" href="#" style="display:none">恢复默认</a>

```

(5) 在页面中插入显示日记图片的标记和面板标记<canvas>，关键代码如下：

```


<canvas id="canvas${id.count }" style="display:none;"></canvas>

```

(6) 在页面的底部，还需要实现当页面载入完成后，通过 while 循环执行旋转图片的方法，具体代码如下：

```

<script type="text/javascript">
window.onload = function(){
    while(i>0){
        eval("rotate"+i)(); //执行旋转图片的方法
        i--;
    }
}
</script>

```

A.6.5 显示全部九宫格日记的实现过程

用户访问清爽夏日九宫格日记网时，进入的页面就是显示全部九宫格日记页面。在该页面将分页显示最新的 50 条九宫格日记，具体的实现过程如下：

(1)编写处理日记信息的 Servlet“DiaryServlet”，在该类中，首先需要在构造方法中实例化 DiaryDao 类（该类用于实现与数据库的交互），然后编写 doGet()和 doPost()方法，在这两个方法中根据 request 的 getParameter()方法获取的 action 参数值执行相应方法，由于这两个方法中的代码相同，所以只需在第一个方法 doPost()中写相应代码，在另一个方法 doGet()中调用 doPost()方法即可。

```
public class DiaryServlet extends HttpServlet {
    MyPagination pagination = null;           // 数据分页类的对象
    DiaryDao dao = null;                     // 日记相关的数据库操作类的对象
    public DiaryServlet() {
        super();
        dao = new DiaryDao();               // 实例化日记相关的数据库操作类的对象
    }
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        String action = request.getParameter("action");
        if ("preview".equals(action)) {
            preview(request, response);       // 预览九宫格日记
        } else if ("save".equals(action)) {
            save(request, response);          // 保存九宫格日记
        } else if ("listAllDiary".equals(action)) {
            listAllDiary(request, response);  // 查询全部九宫格日记
        } else if ("listMyDiary".equals(action)) {
            listMyDiary(request, response);   // 查询我的日记
        } else if ("delDiary".equals(action)) {
            delDiary(request, response);      // 删除我的日记
        }
    }
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        doPost(request, response);           // 执行doPost()方法
    }
}
```

(2) 在处理日记信息的 Servlet “DiaryServlet” 中，编写 action 参数 listAllDiary 对应的方法 listAllDiary()。在该方法中，首先获取当前页码，并判断是否为页面初次运行，如果是初次运行，则调用 Dao 类中的 queryDiary()方法获取日记内容，并初始化分页信息，否则获取当前页面，并获取指定页数据，最后保存当前页的日记信息等，并重定向页面。listAllDiary()方法的具体代码如下：

```
public void listAllDiary(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String strPage = (String) request.getParameter("Page"); // 获取当前页码
    int Page = 1;
    List<Diary> list = null;
```

```

        if (strPage == null) {
            // 当页面初次运行
            String sql = "select d.*,u.username from tb_diary d inner join tb_user u on u.id=d.userid order by d.writeTime DESC limit 50";
            pagination = new MyPagination();
            list = dao.queryDiary(sql);
            // 获取日记内容
            int pagesize = 4;
            // 指定每页显示的记录数
            list = pagination.getInitPage(list, Page, pagesize);
            // 初始化分页信息
            request.getSession().setAttribute("pagination", pagination);
        } else {
            pagination = (MyPagination) request.getSession().getAttribute("pagination");
            Page = pagination.getPage(strPage);
            // 获取当前页码
            list = pagination.getAppointPage(Page);
            // 获取指定页数据
        }
        request.setAttribute("diaryList", list);
        // 保存当前页的日记信息
        request.setAttribute("Page", Page);
        // 保存的当前页码
        request.setAttribute("url", "listAllDiary");
        // 保存当前页面的URL
        request.getRequestDispatcher("listAllDiary.jsp").forward(request, response);
        // 重定向页面
    }
}

```

(3) 在对日记进行操作的 `DiaryDao` 类中，编写用于查询日记信息的方法 `queryDiary()`，在该方法中，首先执行查询语句，然后应用 `while` 循环将获取的日记信息保存到 `List` 集合中，最后返回该 `List` 集合，具体代码如下：

```

public List<Diary> queryDiary(String sql) {
    ResultSet rs = conn.executeQuery(sql);
    // 执行查询语句
    List<Diary> list = new ArrayList<Diary>();
    try {
        // 捕获异常
        while (rs.next()) {
            Diary diary = new Diary();
            diary.setId(rs.getInt(1));
            // 获取并设置ID
            diary.setTitle(rs.getString(2));
            // 获取并设置日记标题
            diary.setAddress(rs.getString(3));
            // 获取并设置图片地址
            Date date;
            try {
                date = DateFormat.getDateInstance().parse(rs.getString(4));
                diary.setWriteTime(date);
                // 设置写日记的时间
            } catch (ParseException e) {
                e.printStackTrace();
                // 输出异常信息到控制台
            }
            diary.setUserid(rs.getInt(5));
            // 获取并设置用户ID
            diary.setUsername(rs.getString(6));
            // 获取并设置用户名
            list.add(diary);
            // 将日记信息保存到list集合中
        }
    } catch (SQLException e) {
        e.printStackTrace();
        // 输出异常信息
    } finally {
        conn.close();
        // 关闭数据库连接
    }
}

```



```

    }
    return list;
}

```

(4) 编写 listAllDiary.jsp 文件，用于分页显示全部九宫日记，具体的实现过程如下：

引用 JSTL 的核心标签库和格式与国际化标签库，并应用<jsp:useBean>指令引入保存分页代码的 JavaBean “MyPagination”，具体代码如下：

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<jsp:useBean id="pagination" class="com.wgh.tools.MyPagination" scope="session"/>

```

应用 JSTL 的<c:if>标签判断是否存在日记列表，如果存在，则应用 JSTL 的<c:forEach>标签循环显示指定条数的日记信息。具体代码如下：

```

<c:if test="${!empty requestScope.diaryList}">
<c:forEach items="${requestScope.diaryList}" var="diaryList" varStatus="id">
    <div style="border-bottom-color:#CBCBCB;padding:5px;border-bottom-style:dashed;border-bottom-width:
1px;margin: 10px 20px;color:#0F6548">
        <font color="#CE6A1F" style="font-weight: bold;font-size:14px;">${diaryList.username}</font>&nbsp;&nbsp;&nbsp;发
表九宫格日记: <b>${diaryList.title}</b></div>
        <div style="margin:10px 10px 0px 10px;background-color:#FFFFFF;
border-bottom-color:#CBCBCB;border-bottom-style:dashed;border-bottom-width: 1px;">
            <div id="diaryImg${id.count }" style="border:1px #dddddd solid;width:60px;background-color:#EEEEEE;">
                <div id="control${id.count }" style="display:none;padding: 10px;">
                    <%String url=request.getRequestURL().toString();
url=url.substring(0,url.lastIndexOf("/"));%>
                    <a href="#" onClick="zoom('${id.count }','${diaryList.address }')">收缩</a>&nbsp;&nbsp;&nbsp;
                    <a href="<%=url %>/images/diary/${diaryList.address }.png" target="_blank">查看原图</a>
                    &nbsp;&nbsp;&nbsp;<a id="rotLeft${id.count }" href="#">左转</a>
                    &nbsp;&nbsp;&nbsp;<a id="rotRight${id.count }" href="#">右转</a>
                    <a id="reDefault${id.count }" href="#" style="display:none">恢复默认</a>
                </div>
                
                <canvas id="canvas${id.count }" style="display:none;" onClick="zoom('${id.count }','${diaryList.address }')">
                </canvas>
            </div>
            <div style="padding:10px;background-color:#FFFFFF;text-align:right;color:#999999;">
                发表时间: <fmt:formatDate value="${diaryList.writeTime}" type="both" pattern="yyyy-MM-dd
HH:mm:ss"/>
                <c:if test="${sessionScope.userName==diaryList.username}">
                    <a
href="DiaryServlet?action=delDiary&id=${diaryList.id }&url=${requestScope.url}&imgName=${diaryList.address }">[ 删
除]</a>
                </c:if>
            </div>
        </div>
    </div>

```

```
</c:forEach>
</c:if>
```

应用 JSTL 的<c:if>标签判断是否存在日记列表,如果不存在,则显示提示信息“暂无九宫格日记!”。具体代码如下:

```
<c:if test="${empty requestScope.diaryList}">
    暂无九宫格日记!
</c:if>
```

在页面的底部添加分页控制导航栏,具体代码如下:

```
<div style="background-color: #FFFFFF;">
    <%=pagination.printCtrl(Integer.parseInt(request.getAttribute("Page").toString()),"DiaryServlet?action="+request.getAtt
    ribute("url"),"")%>
</div>
```

A.6.6 我的日记的实现过程

用户注册并成功登录到清爽夏日九宫格日记网后,就可以查看自己的日记。例如,用户 wgh 登录后,单击导航栏中的“我的日记”超级链接,将显示如图 A.21 所示的运行结果。



图 A.21 我的日记的运行结果

由于我的日记功能和显示全部九宫格日记功能的实现方法类似，所不同的是查询日记内容的 SQL 语句不同，所以在本网站中，我们将操作数据库所用的 Dao 类及显示日记列表的 JSP 页面使用同一个。下面我们就给出在处理日记信息的 Servlet “DiaryServlet” 中，查询我的日记功能所需要的 action 参数 listMyDiary 对应的方法的具体内容。

在该方法中，首先获取当前页码，并判断是否为页面初次运行，如果是初次运行，则调用 Dao 类中的 queryDiary() 方法获取日记内容（此时需要应用内联接查询对应的日记信息），并初始化分页信息，否则获取当前页面，并获取指定页数据，最后保存当前页的日记信息等，并重定向页面。listMyDiary() 方法的具体代码如下：

```
private void listMyDiary(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    String strPage = (String) request.getParameter("Page");           // 获取当前页码
    int Page = 1;
    List<Diary> list = null;
    if (strPage == null) {
        int userid = Integer.parseInt(session.getAttribute("uid"))
            .toString();                                               // 获取用户ID号
        String sql = "select d.*,u.username from tb_diary d inner join tb_user u on u.id=d.userid where d.userid="
            + userid + " order by d.writeTime DESC";                  // 应用内联接查询日记信息
        pagination = new MyPagination();
        list = dao.queryDiary(sql);                                    // 获取日记内容
        int pagesize = 4;                                             // 指定每页显示的记录数
        list = pagination.getInitPage(list, Page, pagesize);          // 初始化分页信息
        request.getSession().setAttribute("pagination", pagination); // 保存分页信息
    } else {
        pagination = (MyPagination) request.getSession().getAttribute(
            "pagination");                                             // 获取分页信息
        Page = pagination.getPage(strPage);
        list = pagination.getAppointPage(Page);                       // 获取指定页数据
    }
    request.setAttribute("diaryList", list);                          // 保存当前页的日记信息
    request.setAttribute("Page", Page);                               // 保存的当前页码
    request.setAttribute("url", "listMyDiary");                       // 保存当前页的URL地址
    request.getRequestDispatcher("listAllDiary.jsp").forward(request, response); // 重定向页面到listAllDiary.jsp
}
```

A.6.7 删除我的日记的实现过程

用户注册并成功登录到清爽夏日九宫格日记网后，就可以删除自己发表的日记。在删除日记时，不仅将数据库中对应的记录删除，而且将服务器中保存的日记图片也一起删除，下面介绍具体的实现过程。

(1) 在处理日记信息的 Servlet “DiaryServlet” 中，编写 action 参数 delDiary 对应的方法 delDiary()。在该方法中，首先获取要删除的日记信息，并调用 DiaryDao 类的 delDiary() 方法从数据表中删除日记信息，然后判断删除日记是否成功，如果成功再删除对应日记的图片和缩略图，并弹出删除日记成功

的提示对话框，否则弹出删除日记失败的提示对话框。`delDiary()`方法的具体代码如下：

```
private void delDiary(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));           // 获取要删除的日记的ID
    String imgName = request.getParameter("imgName");               // 获取图片名
    String url = request.getParameter("url");                       // 获取返回的URL地址
    int rtn = dao.delDiary(id);                                     // 删除日记
    PrintWriter out = response.getWriter();
    if (rtn > 0) {                                                  // 当删除日记成功时
        /***** 删除日记图片及缩略图 *****/
        String path = getServletContext().getRealPath("/") + "images\\diary\\";
        java.io.File file = new java.io.File(path + imgName + "scale.jpg"); // 获取缩略图
        file.delete();                                              // 删除指定的文件
        file = new java.io.File(path + imgName + ".png");          // 获取日记图片
        file.delete();                                              // 删除指定的文件
        /*****/
        out
            .println("<script>alert('删除日记成功！');window.location.href='DiaryServlet?action="
                + url + "';</script>");
    } else {                                                        // 当删除日记失败时
        out
            .println("<script>alert('删除日记失败，请稍后重试！');history.back();</script>");
    }
}
```

(2) 在对日记进行操作的 `DiaryDao` 类中，编写用于删除日记信息的方法 `delDiary()`，在该方法中，首先编写删除数据所用的 SQL 语句，然后执行该语句，最后返回执行结果，具体代码如下：

```
public int delDiary(int id) {
    String sql = "DELETE FROM tb_diary WHERE id=" + id;
    int ret = 0;
    try {
        ret = conn.executeUpdate(sql); // 执行更新语句
    } catch (Exception e) {
        e.printStackTrace();          // 输出异常信息
    } finally {
        conn.close();                 // 关闭数据连接
    }
    return ret;
}
```

A.7 写九宫格日记模块设计

A.7.1 写九宫格日记概述

用户注册并成功登录到清爽夏日九宫格日记网后，就可以写九宫格日记了。写九宫格日记主要由填写日记信息、预览生成的日记图片和保存日记图片 3 部分组成。写九宫格日记的基本流程如图 A.22 所示。

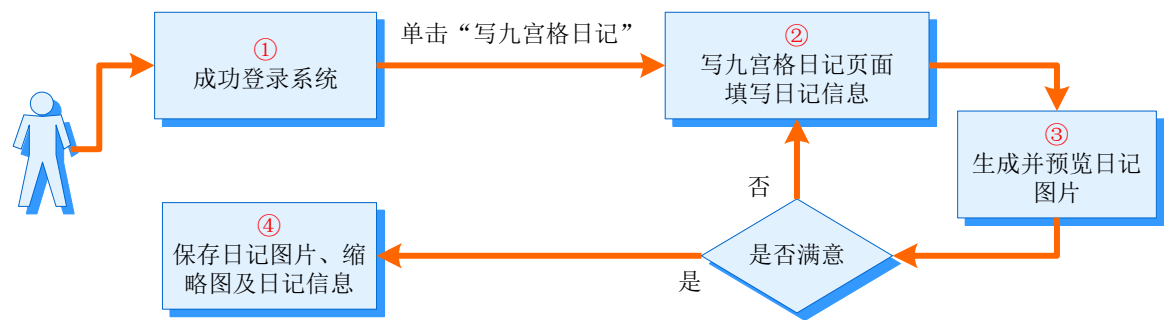


图 A.22 写九宫格日记的基本流程

A.7.2 应用 JQuery 让 PNG 图片在 IE 6 下背景透明

在网页中，常用的可以将背景设置为透明的图片格式有 GIF 和 PNG 两种。不过 GIF 格式的图片质量相对差些，有时在图片的边缘会有锯齿，这时就需要使用 PNG 格式的图片。默认情况下，IE 6 浏览器不支持 PNG 图片的背景透明（当网页中插入背景透明的 PNG 图片时，其背景将带有蓝灰色的背景，如图 A.23 所示），而 IE 7 和 IE 8 就可以支持（运行效果如图 A.24 所示）。考虑到现在还有很多人在使用 IE 6 浏览器，所以需要通过编码解决这一问题。




图 A.23 IE 6 下 PNG 图片背景不透明的效果



图 A.24 IE 8 下 PNG 图片背景透明的效果

解决 PNG 图片在 IE 6 下背景不透明的问题，可以使用 JQuery 及其 pngFix 插件实现。下面介绍具体的实现过程。

 **说明：** JQuery 的 pngFix 插件用于让 IE 5.5 和 IE 6 下 PNG 图片背景透明。

(1) 下载 JQuery 和 pngFix 插件。本项目中使用的 JQuery 是下载 pngFix 插件时带的 jquery-A.3.2.min.js，并没有单独下载。pngFix 插件的下载地址是：
<http://jquery.adnreaseberhard.de/download/pngFix.zip>。

(2) 下载 pngFix 插件后，将得到一个名称为 pngFix.zip 文件，解压缩该文件后，可以得到 jquery-21.3.2.min.js、jquery.pngFix.js 和 jquery.pngFix.pack.js3 个 JS 文件，将这 3 个文件复制到项目的 JS 文件夹中，然后在需要将 PNG 图片设置为背景透明的页面中包含这 3 个文件，具体代码如下：

```
<script type="text/javascript" src="JS/jquery-21.3.2.min.js"></script>
<script type="text/javascript" src="JS/pluginpage.js"></script>
<script type="text/javascript" src="JS/jquery.pngFix.pack.js"></script>
```

(3) 在页面的<head>标记中编写 JQuery 代码，使用 pngFix 插件，具体代码如下：

```
<script type="text/javascript">
    $(document).ready(function(){
        $('div.examples').pngFix( );
    });
</script>
```

(4) 将要显示的 PNG 图片应用<div>标记括起来，该 div 标记使用类选择器 examples 定义的样式，关键代码如下：

```
<div class="examples">
    <!--插入PNG图片的代码-->
</div>
```

A.7.3 填写日记信息的实现过程

用户成功登录到清爽夏日九宫格日记网后，单击导航栏中的“写九宫格日记”超级链接，将进入到填写日记信息的页面，在该页面中，用户可选择日记模板，单击某个模板标题时，将在下方给出预览效果，选择好要使用的模板后（这里选择“女孩”模板），就可以输入日记标题（这里为“心情很好”），接下来就是通过在九宫格中填空来实现日记的编写了，一些都填写好后（如图 A.25 所示），就可以单击“预览”按钮，预览完成效果。

请选择模板: 默认

请输入日记标题:

开心的事

- ☐ 工作完成了
- ☐ 我还活着
- ☐ 瘦了
- ☐ 好多好吃的

为他人做的事

- ☐ 关心同事
- ☐ 问候家人了
- ☐ 给老人让坐
- ☐ 忘记了

工作/计划/备忘

- ☐ 写工作总结
- ☐ 出去旅游
- ☐ 继续努力工作
- ☐ 休息一下

比起昨天的进步

- ☐ 效率提高了
- ☐ 看书了
- ☐ 状态好了
- ☐ 不再空想了

2011年6月9日 星期四

心情/感悟/灵感

- ☐ 心情不错
- ☐ 不给力
- ☐ 幸福
- ☐ 神马都是浮云

关注/新闻/八卦

- ☐ 她·他写九宫格日记了
- ☐ 白菜贵了
- ☐ 大家都在关注神马
- ☐ 新闻联播

健康/体重/饮食

- ☐ 瘦了
- ☐ 胖了
- ☐ 健康饮食
- ☐ 一日三餐不能少

梦/梦想

- ☐ 睡得很好
- ☐ 拥有自己的房子
- ☐ 忘记了
- ☐ 努力做好自己

预览

图 A.25 填写九宫格日记页面

(1) 编写填写九宫格日记的文件 `writeDiary.jsp`, 在该文件中添加一个用于收集日记信息的表单, 具体代码如下:

```
<form name="form1" method="post" action="DiaryServlet?action=preview">
</form>
```

(2) 在上面的表单中, 首先添加一个用于设置模板的 `<div>` 标记, 并在该 `<div>` 标记中添加 3 个用于设置模板的超级链接和一个隐藏域, 用于记录所选择的模板, 然后再添加一个用于填写日记标题的 `<div>` 标记, 并在该 `<div>` 标记中添加一个文本框, 用于填写日记标题, 具体代码如下:

```

<div style="margin:10px;"><span class="title">请选择模板: </span><a href="#" onClick="setTemplate('默认')">默认
</a> <a href="#" onClick="setTemplate('女孩')">女孩</a> <a href="#" onClick="setTemplate('怀旧')">怀旧</a>
<input id="template" name="template" type="hidden" value="默认">
</div>
<div style="padding:10px;" class="title">请输入日记标题: <input name="title" type="text" size="30" maxlength="30"
value="请在此输入标题" onFocus="this.select()"></div>

```

(3) 编写用于预览所选择模板的 JavaScript 自定义函数 setTemplate(), 在该函数中引用的 writeDiary_bg 元素, 将在步骤(4)中进行添加。setTemplate()函数的具体代码如下:

```

function setTemplate(style){
    if(style=="默认"){
        document.getElementById("writeDiary_bg").style.backgroundImage="url(images/diaryBg_00.jpg)";
        document.getElementById("writeDiary_bg").style.width="738px";           //宽度
        document.getElementById("writeDiary_bg").style.height="751px";           //高度
        document.getElementById("writeDiary_bg").style.paddingTop="50px";         //顶边距
        document.getElementById("writeDiary_bg").style.paddingLeft="53px";        //左边距
        document.getElementById("template").value="默认";
    }else if(style=="女孩"){
        document.getElementById("writeDiary_bg").style.backgroundImage="url(images/diaryBg_021.jpg)";
        document.getElementById("writeDiary_bg").style.width="750px";           //宽度
        document.getElementById("writeDiary_bg").style.height="629px";           //高度
        document.getElementById("writeDiary_bg").style.paddingTop="160px";        //顶边距
        document.getElementById("writeDiary_bg").style.paddingLeft="50px";        //左边距
        document.getElementById("template").value="女孩";
    }else{
        document.getElementById("writeDiary_bg").style.backgroundImage="url(images/diaryBg_02.jpg)";
        document.getElementById("writeDiary_bg").style.width="740px";           //宽度
        document.getElementById("writeDiary_bg").style.height="728px";           //高度
        document.getElementById("writeDiary_bg").style.paddingTop="30px";         //顶边距
        document.getElementById("writeDiary_bg").style.paddingLeft="60px";        //左边距
        document.getElementById("template").value="怀旧";
    }
}

```

(4) 添加一个用于设置日记背景的<div>标记, 并将标记的 id 属性设置为 writeDiary_bg, 关键代码如下:

```

<div id="writeDiary_bg">
    <!--此处省略了设置日记内容的九宫格代码-->
</div>

```

(5) 编写 CSS 代码, 用于控制日记背景, 关键代码如下:

```

#writeDiary_bg{                               /*设置日记背景的样式*/
width:738px;                                  /*设置宽度*/
height:751px;                                 /*设置高度*/
background-repeat:no-repeat;                  /*设置背景不重复*/
background-image:url(images/diaryBg_00.jpg); /*设置默认的背景图片*/
padding-top:50px;                             /*设置顶边距*/
}

```



```
padding-left:53px;                /*设置左边距*/
}
```

(6) 在 id 为 writeDiary_bg 的<div>标记中添加一个宽度和高度都是 600 的<div>标记, 用于添加以九宫格方式显示日记内容的无序列表, 关键代码如下:


```
<div style="width:600px; height:600px; ">
</div>
```

(7) 在步骤(6)中添加的<div>标记中添加一个包含 9 个列表项的无序列表, 用于布局显示日记内容的九宫格。关键代码如下:

```
<ul id="gridLayout">
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
</ul>
```

(8) 编写 CSS 代码, 控制上面的无序列表的显示样式, 让其每行显示 3 个列表项, 具体代码如下:

```
#gridLayout {                /*设置写日记的九宫格的<ul>标记的样式*/
    float: left;              /*设置浮动方式*/
    list-style: none;         /*不显示项目符号*/
    width: 100%;              /*设置宽度为100%*/
    margin: 0px;              /*设置外边距*/
    padding: 0px;             /*设置内边距*/
    display: inline;          /*设置显示方式*/
}
#gridLayout li {              /*设置写日记的九宫格的<li>标记的样式*/
    width: 33%;               /*设置宽度*/
    float: left;              /*设置浮动方式*/
    height: 198px;            /*设置高度*/
    padding: 0px;             /*设置内边距*/
    margin: 0px;              /*设置外边距*/
    display: inline;          /*设置显示方式*/
}
```

 说明: 通过 CSS 控制的无序列表的显示样式如图 A.26 所示, 其中, 该图中的边框线在网站运行时是没有的, 这是为了让读者看到效果而后设置的。

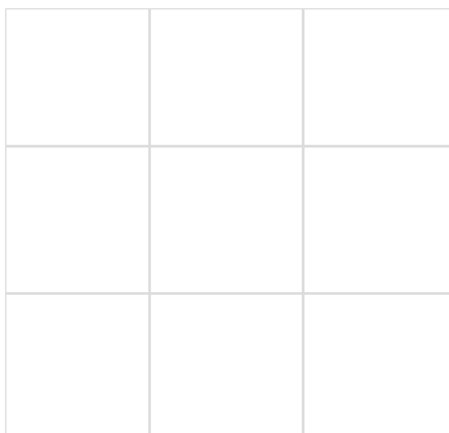


图 A.26 通过 CSS 控制后的无序列表的显示效果

(9) 在图 A.26 所示的九宫格的每个格子中添加用于填写日记内容的文本框及预置的日记内容。由于在这个九宫格中，除了中间的那个格子外（即第 5 个格子），其他的 8 个格子的实现方法是相同，所以这里将以第一个格子为例进行介绍。

添加一个用于设置内容的<div>标记，并使用自定义的样式选择器 `cssContent`，关键代码如下：

```
<style>
.cssContent{                                /*设置内容的样式*/
    float:left;
    padding:40px 0px;                       /*设置上、下内边距为40，左、右内边距为0*/
    display:inline;                         /*设置显示方式*/
}
</style>
<div class="cssContent"></div>
```

在上面的<div>标记中，添加一个包含 5 个列表项的无序列表，其中，第一个列表项中添加一个文本框，其他 4 个设置预置内容，关键代码如下：

```
<ul id="opt">
    <li>
        <input name="content" type="text" size="30" maxlength="15" value="请在此输入文字" onFocus="this.select()">
    </li>
    <li>
        <a href="#" onClick="document.getElementsByName('content')[0].value='工作完成了'">◎ 工作完成了</a>
    </li>
    <li><a href="#" onClick="document.getElementsByName('content')[0].value='我还活着'">◎ 我还活着</a></li>
    <li><a href="#" onClick="document.getElementsByName('content')[0].value='瘦了'">◎ 瘦了</a></li>
    <li>
        <a href="#" onClick="document.getElementsByName('content')[0].value='好多好吃的'">◎ 好多好吃的</a>
    </li>
</ul>
```

 **技巧:** 在本项目中, 共设置了 9 个名称为 content 的文本框, 用于以控件数组的方式记录日记内容。

这样, 当表单被提交后, 在服务器中就可以应用 request 对象的 getParameterValues() 方法来获取字符串数组形式的日记内容, 比较方便。

编写 CSS 代码, 用于控制列表项的样式, 具体代码如下:

```
#opt{                                /*设置默认选项相关的<ul>标记的样式 */
    padding:0px 0px 0px 10px;        /*设置上、右、下内边距为0, 左内边距为10*/
    margin:0px;                      /*设置外边距*/
}
#opt li{                             /*设置默认选项相关的<li>标记的样式 */
    width:99%;
    padding-top:5px 0px 0px 10px;
    font-size:14px;                  /*设置字体大小为14像素*/
    height:25px;                     /*设置高度*/
    clear:both;                      /*左、右两侧不包含浮动内容*/
}
```

(10) 实现九宫格的中间的那个格子, 也就是第 5 个格子, 该格子用于显示当前日期和天气, 具体代码如下:

```
<ul id="weather"><li style="height:27px;"> <span id="now" style="font-size: 14px;font-weight:bold;padding-left:5px;">
正在获取日期</span>
    <input name="content" type="hidden" value="weathervalue"><br></br>
    <div class="examples">
        <input name="weather" type="radio" value="1">
        
        <input name="weather" type="radio" value="2">
        
        <input name="weather" type="radio" value="3">
        
        <input name="weather" type="radio" value="4">
        
        <input name="weather" type="radio" value="5" checked="checked">
        
        <input name="weather" type="radio" value="6">
        
        <input name="weather" type="radio" value="7">
        
        <input name="weather" type="radio" value="8">
        
        <input name="weather" type="radio" value="9">
        
    </div>
    </li>
</ul>
```

(11) 编写 JavaScript 代码, 用于在页面载入后, 获取当前日期和星期, 显示到 id 为 now 的标记中, 具体代码如下:

```

window.onload=function(){
    var date=new Date();           //创建日期对象
    year=date.getFullYear();       //获取当前日期中的年份
    month=date.getMonth();        //获取当前日期中的月份
    day=date.getDate();           //获取当时日期中的日
    week=date.getDay();           //获取当前日期中的星期
    var arr=new Array("星期日","日期一","星期二","星期三","星期四","星期五","星期六");
    document.getElementById("now").innerHTML=year+"年"+(month+1)+"月"+day+"日 "+arr[week];
}

```

(12) 在 id 为 writeDiary_bg 的<div>标记后面添加一个<div>标记，并在该标记中添加一个提交按钮，用于显示预览按钮，具体代码如下：

```
<div style="height:30px;padding-left:360px;"><input type="submit" value="预览"></div>
```

A.7.4 预览生成的日记图片的实现过程

用户在填写日记信息页面填写好日记信息后，就可以单击“预览”按钮，预览完成的效果，如图 A.27 所示。如果感觉日记内容不是很满意，可以单击“再改改”超级链接，进行修改，否则可以单击“保存”超级链接保存该日记。



图 A.27 预览生成的日记图片

(1) 在处理日记信息的 Servlet “DiaryServlet” 中，编写 action 参数 preview 对应的方法 preview()。在该方法中，首先获取日记标题、日记模板、天气和日记内容，然后将为没有设置内容的项目设置默认值，最后保存相应信息到 session 中，并重定向页面到 preview.jsp。preview() 方法的具体代码如下：

```
public void preview(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String title = request.getParameter("title");           // 获取日记标题
    String template = request.getParameter("template");     // 获取日记模板
    String weather = request.getParameter("weather");       // 获取天气
    String[] content = request.getParameterValues("content"); // 获取日记内容
    for (int i = 0; i < content.length; i++) {              // 为没有设置内容的项目设置默认值
        if (content[i].equals(null) || content[i].equals("") || content[i].equals("请在此输入文字")) {
            content[i] = "没啥可说的";
        }
    }
    HttpSession session = request.getSession(true);        // 获取HttpSession
    session.setAttribute("template", template);             // 保存选择的模板
    session.setAttribute("weather", weather);               // 保存天气
    session.setAttribute("title", title);                   // 保存日记标题
    session.setAttribute("diary", content);                 // 保存日记内容
}
```

```
request.getRequestDispatcher("preview.jsp").forward(request, response);    // 重定向页面
}
```

(2) 编写 `preview.jsp` 文件, 在该文件中, 首先显示保存到 `session` 中的日记标题, 然后添加预览日记图片的 `` 标记, 并将其 `id` 属性设置为 `diaryImg`, 关键代码如下:

[illegible]

(3)为了让页面载入后,再显示预览图片,还需要编写 JavaScript 代码,设置 id 为 diaryImg 的标记的图片来源,这里指定的是一个 Servlet 映射地址。关键代码如下:

```
<script language="javascript">
window.onload=function(){
    document.getElementById("diaryImg").src="CreateImg";
}
</script>
```

(4)编写用于生成预览图片的Servlet,名称为CreateImg,该类继承HttpServlet,主要通过service()方法生成预览图片,具体的实现过程如下:

创建 Servlet “CreateImg”，并编写 service()方法，在该方法中，首先指定生成的响应是图片，以及图片的宽度和高度，然后获取日记模板、天气和图片的完整路径，再根据选择的模板绘制背景图片及相应的日记内容，最后输出生成的日记图片，并保存到 Session 中，具体代码如下：

```

public class CreateImg extends HttpServlet {
    public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        response.setHeader("Pragma", "No-cache");           // 禁止缓存
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/jpeg");              // 指定生成的响应是图片
        int width = 600;                                     // 图片的宽度
        int height = 600;                                    // 图片的高度
        BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                   // 获取Graphics类的对象
        HttpSession session = request.getSession(true);
        String template = session.getAttribute("template"); // 获取模板
        String weather = session.getAttribute("weather");   // 获取天气
        weather = request.getRealPath("images/" + weather + ".png"); // 获取图片的完整路径
        String[] content = (String[]) session.getAttribute("diary");
        File bgImgFile;                                     //背景图片
    }
}

```

```

        if ("默认".equals(template)) {
            bgImgFile = new File(request.getRealPath("images/bg_00.jpg"));
            Image src = ImageIO.read(bgImgFile);           // 构造Image对象
            g.drawImage(src, 0, 0, width, height, null);     // 绘制背景图片
            outWord(g, content, weather, 0, 0);
        } else if ("女孩".equals(template)) {
            bgImgFile = new File(request.getRealPath("images/bg_021.jpg"));
            Image src = ImageIO.read(bgImgFile);           // 构造Image对象
            g.drawImage(src, 0, 0, width, height, null);     // 绘制背景图片
            outWord(g, content, weather, 25, 110);
        } else {
            bgImgFile = new File(request.getRealPath("images/bg_02.jpg"));
            Image src = ImageIO.read(bgImgFile);           // 构造Image对象
            g.drawImage(src, 0, 0, width, height, null);     // 绘制背景图片
            outWord(g, content, weather, 30, 5);
        }
        ImageIO.write(image, "PNG", response.getOutputStream());
        session.setAttribute("diaryImg", image);           // 将生成的日记图片保存到Session中
    }
}

```

在 service()方法的下面编写 outWord()方法，用于将九宫格日记的内容写到图片上，具体代码如下：

```

public void outWord(Graphics g, String[] content, String weather, int offsetX, int offsetY) {
    Font mFont = new Font("微软雅黑", Font.PLAIN, 26);    // 通过Font构造字体
    g.setFont(mFont);                                       // 设置字体
    g.setColor(new Color(0, 0, 0));                        // 设置颜色为黑色
    int contentLen = 0;
    int x = 0;                                              // 文字的横坐标
    int y = 0;                                              // 文字的纵坐标
    for (int i = 0; i < content.length; i++) {
        contentLen = content[i].length();                  // 获取内容的长度
        x = 45 + (i % 3) * 170 + offsetX;
        y = 130 + (i / 3) * 140 + offsetY;
        //判断当前内容是否为天气，如果是天气，则先获取当前日记，并输出，然后再绘制天气图片。
        if (content[i].equals("weathervalue")) {
            File bgImgFile = new File(weather);
            mFont = new Font("微软雅黑", Font.PLAIN, 14);  // 通过Font构造字体
            g.setFont(mFont);                                // 设置字体
            Date date = new Date();
            String newTime = new SimpleDateFormat("yyyy年M月d日 E").format(date);
            g.drawString(newTime, x - 12, y - 60);
            Image src;
            try {
                src = ImageIO.read(bgImgFile);
                g.drawImage(src, x + 10, y - 40, 80, 80, null); // 绘制天气图片
            } catch (IOException e) {
                e.printStackTrace();
            }
            // 构造Image对象
        }
        continue;
    }
}

```

```

    }
    //根据文字的个数控制输出文字的大小。
    if (contentLen < 5) {
        switch (contentLen % 5) {
            case 1:
                mFont = new Font("微软雅黑", Font.PLAIN, 40);    // 通过Font构造字体
                g.setFont(mFont);                                // 设置字体
                g.drawString(content[i], x + 40, y);
                break;
            case 2:
                mFont = new Font("微软雅黑", Font.PLAIN, 36);    // 通过Font构造字体
                g.setFont(mFont);                                // 设置字体
                g.drawString(content[i], x + 25, y);
                break;
            case 3:
                mFont = new Font("微软雅黑", Font.PLAIN, 30);    // 通过Font构造字体
                g.setFont(mFont);                                // 设置字体
                g.drawString(content[i], x + 20, y);
                break;
            case 4:
                mFont = new Font("微软雅黑", Font.PLAIN, 28);    // 通过Font构造字体
                g.setFont(mFont);                                // 设置字体
                g.drawString(content[i], x + 10, y);
            }
        } else {
            mFont = new Font("微软雅黑", Font.PLAIN, 22);        // 通过Font构造字体
            g.setFont(mFont);                                    // 设置字体
            if (Math.ceil(contentLen / 5.0) == 1) {
                g.drawString(content[i], x, y);
            } else if (Math.ceil(contentLen / 5.0) == 2) {
                // 分两行写
                g.drawString(content[i].substring(0, 5), x, y - 20);
                g.drawString(content[i].substring(5), x, y + 10);
            } else if (Math.ceil(contentLen / 5.0) == 3) {
                // 分三行写
                g.drawString(content[i].substring(0, 5), x, y - 30);
                g.drawString(content[i].substring(5, 10), x, y);
                g.drawString(content[i].substring(10), x, y + 30);
            }
        }
    }
    g.dispose();
}

```

(5) 在 web.xml 文件中，配置用于生成预览图片的 Servlet，关键代码如下：

```

<servlet>
    <description></description>
    <display-name>CreateImg</display-name>
    <servlet-name>CreateImg</servlet-name>

```



```

double temp = 0; // 缩放比例
/***** 计算缩放比例 *****/
double tagSize = 60;
if (old_w > old_h) {
    temp = old_w / tagSize;
} else {
    temp = old_h / tagSize;
}
/*****
new_w = (int) Math.round(old_w / temp); // 计算新图片的宽
new_h = (int) Math.round(old_h / temp); // 计算新图片的高
image = new BufferedImage(new_w, new_h, BufferedImage.TYPE_INT_RGB);
src = src.getScaledInstance(new_w, new_h, Image.SCALE_SMOOTH);
image.getGraphics().drawImage(src, 0, 0, new_w, new_h, null);
ImageIO.write(image, "JPG", new File(scaleImgUrl)); // 保存缩略图文件
*****/
/**** 将填写的日记保存到数据库中 *****/
Diary diary = new Diary();
diary.setAddress(String.valueOf(value)); // 设置图片地址
diary.setTitle(session.getAttribute("title").toString()); // 设置日记标题
diary.setUserid(Integer.parseInt(session.getAttribute("uid").toString())); // 设置用户ID
int rtn = dao.saveDiary(diary); // 保存日记
PrintWriter out = response.getWriter();
if (rtn > 0) { // 当保存成功时
    out.println("<script>alert('保存成功!');window.location.href='DiaryServlet?action=listAllDiary';</script>");
} else { // 当保存失败时
    out.println("<script>alert('保存日记失败，请稍后重试!');history.back();</script>");
}
/*****
}

```

(2) 在对日记进行操作的 `DiaryDao` 类中，编写用于保存日记信息的方法 `saveDiary()`，在该方法中，首先编写执行插入操作的 SQL 语句，然后执行该语句，将日记信息保存到数据库中，再关闭数据库连接，最后返回执行结果。`saveDiary()`方法的具体代码如下：

```

public int saveDiary(Diary diary) {
    String sql = "INSERT INTO tb_diary (title,address,userid) VALUES('"+ diary.getTitle() + "','" +
diary.getAddress() + "','" + diary.getUserid() + "')"; //保存数据的SQL语句
    int ret = conn.executeUpdate(sql); // 执行更新语句
    conn.close(); // 关闭数据库连接
    return ret;
}

```

A.8 项目发布

1. 搭建 Java Web 项目的开发及运行环境。由于笔者在开发项目时应用的开发工具是 Eclipse for Java

EE，所以建议读者也应用 Eclipse for Java EE 来调试并运行该项目。这样可以确保项目正常运行。

2. 项目发布的具体方法。从光盘中拷贝项目文件夹（例如：01），将其存储于您机器中的指定文件夹下，然后按照下面的步骤进行发布和运行。

- ☑ 附加数据库。打开 MySQL 的“MySQL Administrator”，并登录（本系统需要使用 root 和 111 登录），然后单击 restore 节点，在右侧单击“Open backup File”按钮，在弹出的对话框中，选择拷贝到本地机器中的 01\WebContent\Database\db_9griddiary.sql 文件，并单击“打开”按钮。接下来再单击“Open Restore”按钮，即可完成数据库的附加操作。
- ☑ 导入项目。启动 Eclipse for Java EE，在“项目资源管理器”中单击鼠标右键，在弹出的快捷菜单中，选择“导入”/“导入”菜单项，将弹出如图 A.29 所示的“导入”对话框，在该对话框中选择“常规”/“现有项目到工作空间中”节点，单击“下一步”按钮，在弹出的对话框中单击“选择根目录”文本框后面的“浏览”按钮，选择已经拷贝到本地机器中的项目文件夹，单击“完成”按钮即可。

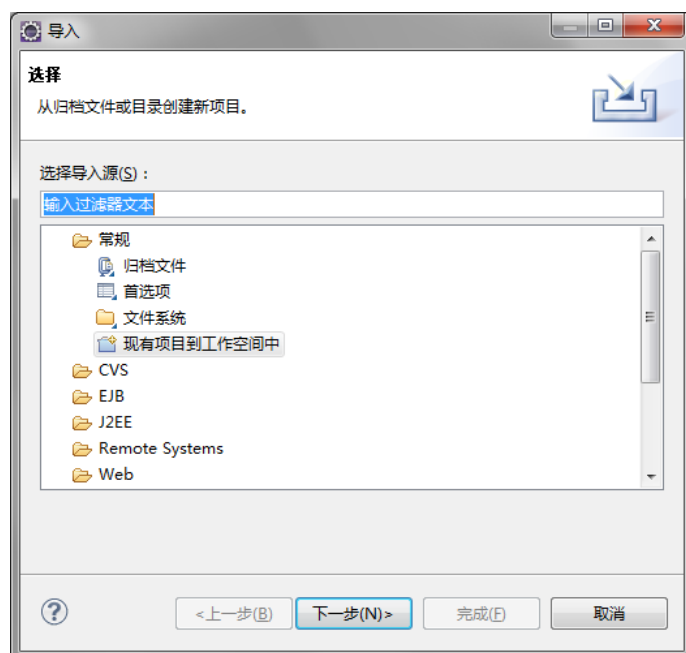


图 A.29 “导入”对话框

- ☑ 运行该项目。在“项目资源管理器”中，展开项目节点，再展开 WebContent 节点，找到 index.jsp 文件，在该文件上单击鼠标右键，在弹出的快捷菜单中选择“运行方式”/“在服务器运行”菜单项，将弹出“在服务器上运行”对话框，在该对话框中，单击“完成”按钮，即可运行该项目。

A.9 小结

在清爽夏日九宫格日记网中，应用到了很多关键的技术，这些技术在开发过程中都是比较常用的技术。下面将简略的介绍一下这些关键技术在实际项目开发中的用处，希望对读者进行二次开发能有一个提示。

1. 本项目采用了 **DIV+CSS** 布局。现在多数网站都采用 **DIV+CSS** 进行网站布局，采用这种布局方式可以提高页面浏览速度，缩减宽带成本。其中，在本项目中应用了让 **DIV+CSS** 布局的页面内容居中显示的技术，该技术在以后开发 **DIV+CSS** 布局的网站开发中，经常可以被用到。

2. 本项目中用户注册功能是通过 **Ajax** 实现的，读者也可以把它提炼出来，应用到自己开发的其他网站中，这样可以节省不少开发时间，以提高开发效率。

3. 在 **Java Web** 项目中，一个最常见的问题就是中文乱码。通常情况下，可以通过配置过滤器进行解决。本项目中就配置了解决中文乱码的过滤器，该过滤器，同样可以配置在其他的网站中。

4. 本项目中应用了在 **Servlet** 中生成日记图片技术和生成缩略图技术，这些技术还可以用来生成随机的图文验证码。

5. 默认情况下，**IE 6** 浏览器不支持 **PNG** 图片的背景透明，为了让 **PNG** 图片在 **IE 6** 浏览器下背景透明，本项目中应用了 **JQuery** 的 **pngFix** 插件实现。该技术也可以应用到任何需要让 **PNG** 图片背景透明的网站中。

6. 本项目中在显示日记列表时，实现了展开和收缩图片，以及对图片进行左转和右转功能，这些技术也比较实用，通常可以应用到博客或网络相册等网站中。