

基于 SSH2 的明日论坛

技术交流平台是一种以技术交流和会员互动为核心的论坛，在这种论坛上用户可以维护自己的文章。也可以针对其他人的文章发表自己的意见，还可以输入关键字搜索相关的文章。随着 IT 技术更新速度的加快，这种论坛将会成为未来 IT 技术服务的主要载体，因而其前景一片光明。本章将向大家介绍如何通过 Struts 2+Spring+Hibernate 来实现这样一种技术交流平台。

通过阅读本章，您可以：

- ▶▶ 了解技术论坛的开发流程
- ▶▶ 掌握如何搭建 SSH2 的项目环境
- ▶▶ 掌握 JavaScript 面向对象编程
- ▶▶ 掌握如何应用 jQuery
- ▶▶ 掌握如何实现 Hibernate 的模糊查询
- ▶▶ 掌握如何利用 Struts 2 标签分页

B.1 开发背景

随着 Internet 技术的快速发展,人与人之间交流方式逐渐增多。网络视频、网络聊天、博客已成为人们彼此沟通、交流信息的主要方式。此外,为了方便人们在某一专业领域探讨问题和发表意见,Internet 上还出现了各种技术交流平台。在技术交流平台上,人们可以对某一领域提出自己遇到的问题,即发表某一主题,随后,论坛上的其他人会根据自己的学识、经验发表意见或解决问题的方法。

到目前为止,已经有一些著名的技术交流平台,如 JavaEye 和 CSDN 等已经成为开发人员的主要活动社区。在这种形式下,作为专业从事软件开发和软件图书创作的明日公司,为了给公司员工以及广大用户提供技术交流平台,公司决定开发“明日论坛”系统。该系统专门为软件编程人员设计。用户可以在这里发表自己的技术文章,也可以阅读别人的文章。可以通过“搜索答案”的方式搜索一种类型的文章。方便大家的学习和交流。

B.2 系统设计

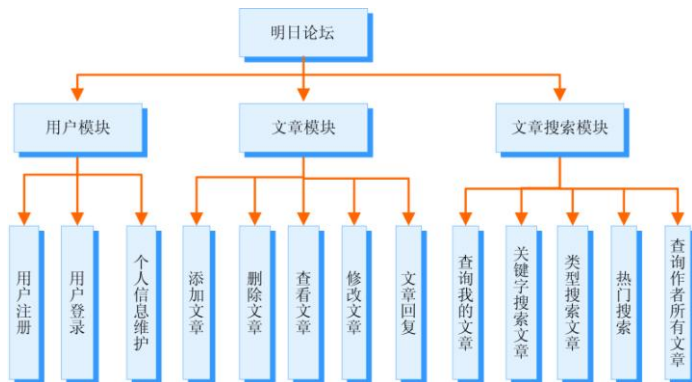
B.2.1 系统目标

对于典型的数据库管理系统,尤其是像论坛这样的数据流量特别大的网络管理系统,必须要满足使用方便、操作灵活等设计需求。本系统在设计时应该满足以下几个目标:

- ☒ 界面友好,采用人机对话方式,操作简单。信息查询灵活、快捷、数据存储安全。
- ☒ 实现用户管理功能,主要包括用户登录与注册功能。
- ☒ 对用户输入的数据,系统进行严格的数据检查,尽可能排除人为错误。
- ☒ 要实现模糊查询功能,允许用户查询一类文章。
- ☒ 全面展示系统内所有分类的帖子。
- ☒ 灵活方便的查询功能。
- ☒ 为用户提供一个方便、快捷的主题信息查看功能。
- ☒ 实现在线发表帖子。
- ☒ 用户随时都可以查看自己发表的帖子,并进行分页显示。
- ☒ 系统运行稳定、安全可靠。

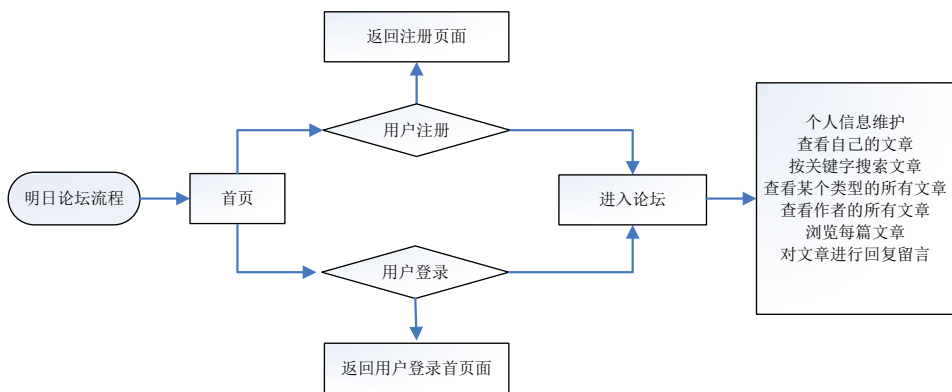
B.2.2 系统功能结构

本论坛主要分为用户模块、文章模块和文章搜索模块 3 大功能模块,用户成功登录后,可以搜索和回复文章,其功能结构如图 B.1 所示。



B.2.3 系统流程图

明日论坛的系统流程图如图 B.2 所示。



B.3 项目开发及运行环境

B.3.1 服务器最低配置

- ☑ CPU: P4 3.2GHz。
- ☑ 内存: 1GB 以上。
- ☑ 硬盘空间: 40GB。
- ☑ 操作系统: Windows 7、Windows XP 或者 Windows 2003。
- ☑ 数据库: MySQL 5.1。
- ☑ Java 开发包: JDK 1.7 以上。
- ☑ Web 服务器: Tomcat 7.0。

- ☑ 开发工具：Eclipse IDE for Java EE。

B.3.2 客户端最低配置

本系统的软件开发环境如下。

- ☑ CPU：赛扬 1.8 以上。
- ☑ 内存：512MB 以上。
- ☑ 显示器：17in 以上显示器。
- ☑ 浏览器：IE 8.0 或者更高版本。
- ☑ 分辨率：1024×768 像素以上。

B.4 系统文件夹组织结构

在开发程序之前，可以把系统中可能用到的文件夹先创建出来（例如：创建一个名为 CSS 的文件夹，用于保存网站中用到的 CSS 样式），这样不仅可以方便以后的程序开发工作，也可以规范网站的整体结构，方便日后的网站维护。在明日论坛系统时，设计了如图 B.3 所示的文件夹架构图。在开发时，只需要将所创建的文件保存在相应的文件夹中就可以了。

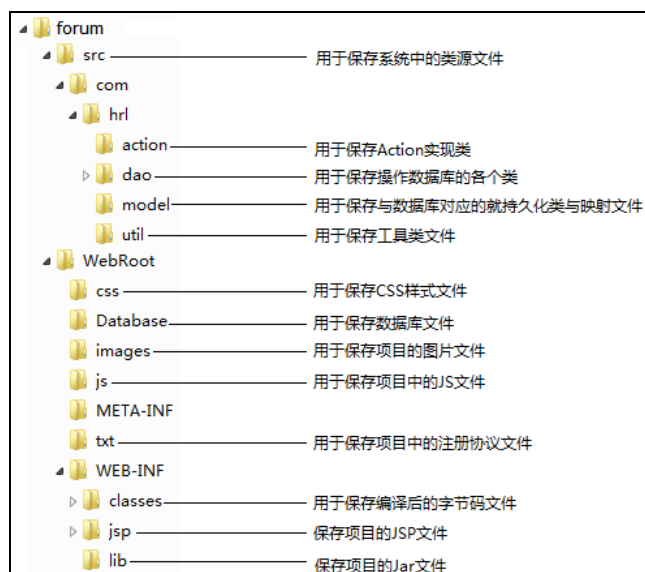


图 B.3 明日论坛的文件夹组织结构

B.5 数据库与数据表设计

本论坛采用 MySQL 作为后台数据库，根据需求分析和功能结构为整个系统涉及 5 个数据表，分

别用于存储用户信息、文章信息、文章类型信息、文章回复信息和文章浏览信息。根据各个表存储的信息和功能，分别设计相应的 E-R 图和数据表。

B.5.1 设计 E-R 图

1. 用户信息表 tb_user 的 E-R 图，如图 B.4 所示。

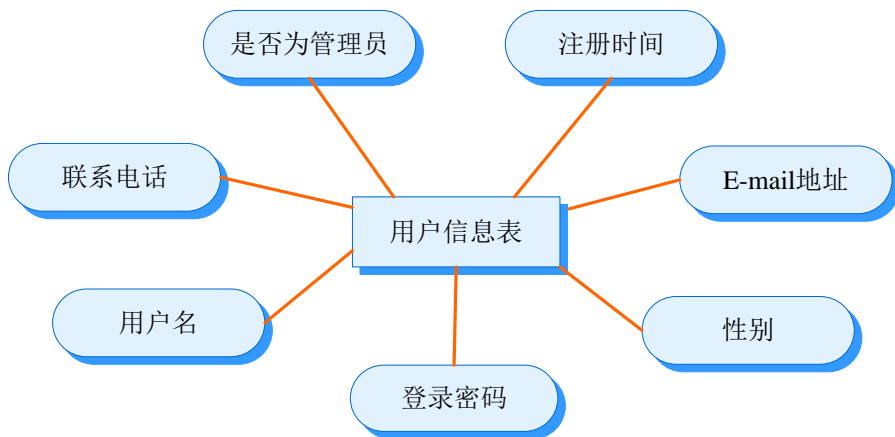


图 B.4 用户信息表 E-R 图

2. 文章信息表 tb_article 的 E-R 图，如图 B.5 所示。

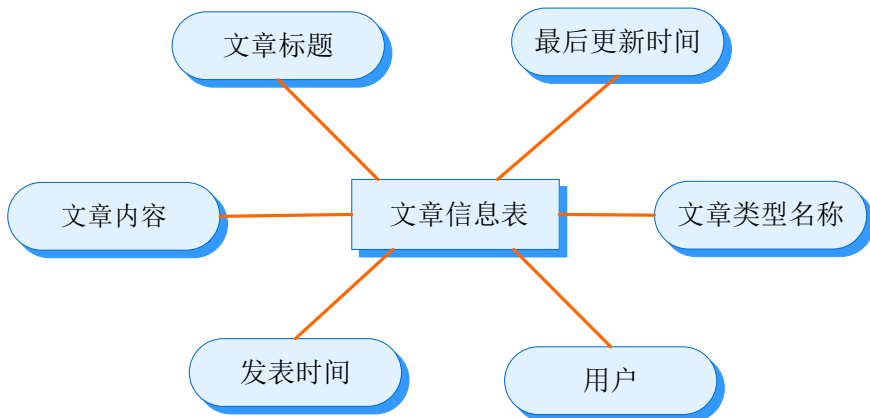


图 B.5 文章信息表的 E-R 图

3. 文章类型信息表 tb_articleType 的 E-R 图，如图 B.6 所示。

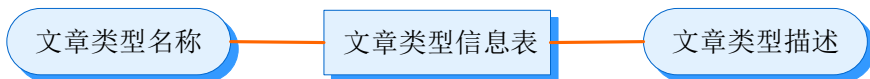


图 B.6 文章类型信息表 E-R 图

4. 文章回复信息表 tb_reply 的 E-R 图，如图 B.7 所示。

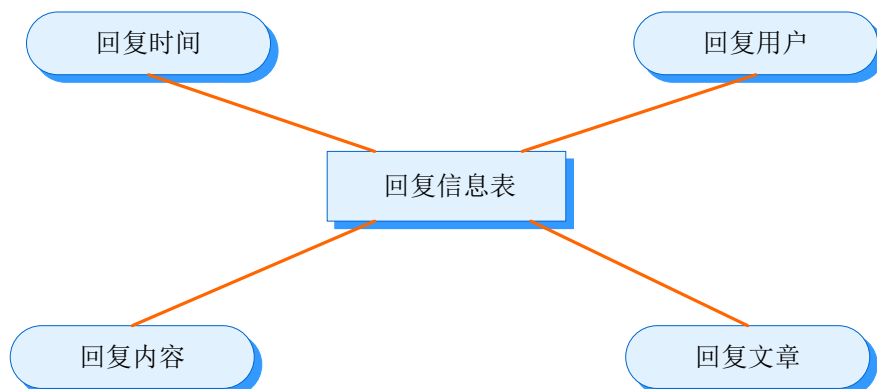


图 B.7 文章回复信息表 E-R 图

4. 文章浏览信息表 tb_scan 的 E-R 图，如图 B.8 所示。

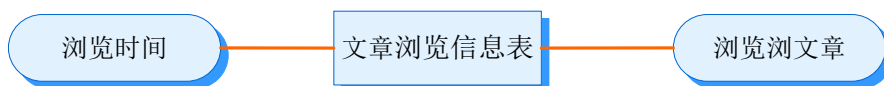


图 B.8 文章浏览信息表 E-R 图

B.5.2 数据库表设计

数据库名为 db_myforum，5 个表的数据表结构如下。

☑ tb_user（用户信息表）：该表用于保存所有的用户信息，其结构如表 B.1 所示。

表 B.1 tb_user 表的结构

| 字 段 名 | 数 据 类 型 | 是 否 为 空 | 是 否 主 键 | 默 认 值 | 说 明 |
|--------------|-------------|---------|---------|-------|---------|
| userId | INT(10) | 否 | 是 | NULL | 系统自动编号 |
| username | VARCHAR(45) | 是 | 否 | NULL | 用户名 |
| password | VARCHAR(45) | 是 | 否 | NULL | 用户登录密码 |
| registerTime | DATETIME | 是 | 否 | NULL | 注册时间 |
| birthday | VARCHAR(20) | 是 | 否 | NULL | 出生年月 |
| email | VARCHAR(45) | 是 | 否 | NULL | 邮箱 |
| Tel | VARCHAR(20) | 是 | 否 | NULL | 联系电话 |
| isAdmin | VARCHAR(2) | 是 | 否 | NULL | 管理员访问次数 |

☑ tb_article（文章信息表）：该表用于保存文章信息，其结构如表 B.2 所示。

表 B.2 tb_article 表的结构

| 字 段 名 | 数 据 类 型 | 是 否 为 空 | 是 否 主 键 | 默 认 值 | 说 明 |
|-----------|---------------|---------|---------|-------|--------|
| articleId | INT(10) | 否 | 是 | NULL | 系统自动编号 |
| title | VARCHAR(255) | 是 | 否 | NULL | 文章标题 |
| content | VARCHAR(2048) | 是 | 否 | NULL | 文章内容 |
| emitTime | DATETIME | 是 | 否 | NULL | 发表时间 |

| | | | | | |
|-----------------|--------------|---|---|------|--------|
| lastUpdateTime | DATETIME | 是 | 否 | NULL | 最后更新时间 |
| articleTypeName | VARCHAR(255) | 是 | 否 | NULL | 文章类型名称 |
| userId | INT(10) | 是 | 否 | NULL | 用户 ID |

☑ **tb_articleType**（文章类型信息表）：该表用于保存所有文章类型信息，其结构如表 B.3 所示。

表 B.3 tb_articleType 表的结构

| 字段名 | 数据类型 | 是否为空 | 是否主键 | 默认值 | 说明 |
|-----------------|--------------|------|------|------|--------|
| articleTypeId | INT(10) | 否 | 是 | NULL | 系统自动编号 |
| articleTypeName | VARCHAR(255) | 否 | 否 | NULL | 文章类型名称 |
| articleTypeDesc | VARCHAR(255) | 是 | 否 | NULL | 文章类型描述 |

☑ **tb_reply**（回复信息表）：该表用于保存所有回复信息，其结果如表 B.4 所示。

表 B.4 tb_reply 表的结构

| 字段名 | 数据类型 | 是否为空 | 是否主键 | 默认值 | 说明 |
|-----------|---------------|------|------|------|--------|
| replyId | INT(10) | 否 | 否 | NULL | 系统自动编号 |
| replyTime | DATETIME | 是 | 是 | NULL | 回复时间 |
| content | VARCHAR(1024) | 是 | 否 | NULL | 回复内容 |
| userId | INT(10) | 是 | 否 | NULL | 用户 ID |
| articleId | INT(10) | 是 | 否 | NULL | 文章 ID |

☑ **tb_scan**（浏览信息表）：该表用于保存所有浏览信息，其结果如表 B.5 所示。

表 B.5 tb_scan 表的结构

| 字段名 | 数据类型 | 是否为空 | 是否主键 | 默认值 | 说明 |
|-----------|----------|------|------|------|---------|
| scanId | INT(10) | 否 | 否 | NULL | 系统自动编号 |
| scanTime | DATETIME | 是 | 是 | NULL | 浏览时间 |
| articleId | INT(10) | 是 | 是 | NULL | 浏览文章 ID |

B.6 公共类设计

将一些常用的操作抽象出来可以提高代码复用率，减少工作量，所以公共类设计的好坏将决定程序整体的开发效率。持久化操作是应用系统中使用频率较高的操作之一，所以常常将程序中的数据库持久化操作方法提取出来，以便随时调用。

B.6.1 Spring+Hibernate 组合实现持久层

由于 Spring 将 Hibernate 集成进来，并对 Hibernate 进行数据源和事务封装，这样我们就可以不用去单独写额外代码管理 Hibernate 的事务处理而把主要精力放在企业级业务逻辑上。关键代码如下：

```
<!-- 配置 sessionFactory -->
<bean id="sessionFactory" class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
```

```

        <property name="configLocation">
            <value>classpath:hibernate.cfg.xml</value>
        </property>
    </bean>
    <!-- 配置事务管理器 -->
    <bean id="transactionManager" class="org.springframework.orm.hibernate4.HibernateTransactionManager">
        <property name="sessionFactory">
            <ref bean="sessionFactory" />
        </property>
    </bean>
    <!-- 配置事务的传播特性 -->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <tx:method name="add*" propagation="REQUIRED" />
            <tx:method name="save*" propagation="REQUIRED" />
            <tx:method name="del*" propagation="REQUIRED" />
            <tx:method name="update*" propagation="REQUIRED" />
            <tx:method name="modify*" propagation="REQUIRED" />
            <tx:method name="*" read-only="true" />
        </tx:attributes>
    </tx:advice>
    <!-- 指定哪些类的哪些方法参与事务 -->
    <aop:config>
        <aop:pointcut id="allManagerMethod" expression="execution(* com.hrl.dao.*.*(..))" />
        <aop:advisor pointcut-ref="allManagerMethod" advice-ref="txAdvice" />
    </aop:config>

```

配置事务和数据源之后，在持久层中获取常用的 Hibernate 方法，其中一些常用方法的代码如下：

```

/ **
 * 保存数据
 * @param object
 * @return
 */
public Serializable save(Object object) {
    return this.getSession().save(object);
}
/ **
 * 删除数据
 * @param clazz
 * @param ids
 */
public void delete(Class clazz, Serializable... ids) {
    Session session=this.getSession();
    for (Serializable id : ids) {
        Object obj = session.load(clazz, id);
        session.delete(obj);
    }
    session.flush();
}


```



```

/**
 * 修改
 */
public void update(Object object) {
    this.getSession().update(object);
}
/**
 * 查询实体的所有对象
 */
public List findAll(Class clazz) {
    return getSession().createQuery("from " + clazz.getName()).list();
}
/**
 * 通过主键加载对象
 */
public Object load(Class clazz, Serializable id) {
    return getSession().load(clazz, id);
}
/**
 * 得到 Criteria 的对象，以方便 QBC 查询
 * @param clazz
 * @return
 */
public Criteria getCriteria(Class clazz){
    return this.getSession().createCriteria(clazz);
}
private SessionFactory sessionFactory;           //定义一个 sessionFactory 对象
/**
 * 获取 Session 对象
 */
protected Session getSession() {
    return sessionFactory.getCurrentSession();    //获取当前 Session
}
public SessionFactory getSessionFactory() {
    return sessionFactory;
}
public void setSessionFactory(SessionFactory sessionFactory) {
    this.sessionFactory = sessionFactory;
}

```

 说明：delete()方法：其参数中的“...”为数组的一种新的写法，相当于 Serializable[] ids。

B.6.2 使用 Struts 2 标签分页

Struts2 对模型驱动支持的很好，它可以再页面上很方便取到业务 Bean 里的属性，同时它的标签库也是非常强大的。鉴于 Struts2 的这些优点，可以将分页也定义成一个可以重用的组件，这将为后续开发省去不少麻烦。

分页页面代码是通过 Struts2 标签来完成，代码如下：

```
<div align="center"><span>每页显示<s:property
    value="page.pageSize" />条</span> <span>共<s:property
    value="page.recordCount" />条</span> <span>当前页<s:property
    value="page.currPage" />共<s:property value="page.pageCount" />页</span> <span>
<s:if test="page.hasPrevious==true">
    <s:a action="%{pageAction}">
    第一页
        <s:param name="page.index" value="0"></s:param>
        <s:param name="page.currPage" value="1"></s:param>
    </s:a>
</s:if><s:else>
    第一页
        </s:else> </span> <span> <s:if test="page.hasPrevious==true">
        <s:a action="%{pageAction}">
        上一页
            <s:param name="page.index" value="page.previousIndex"></s:param>
            <s:param name="page.currPage" value="page.currPage-1"></s:param>
        </s:a>
</s:if><s:else>
        上一页
            </s:else> </span> <span> <s:if test="page.hasNext==true">
            <s:a action="%{pageAction}">
            下一页
                <s:param name="page.index" value="page.nextIndex"></s:param>
                <s:param name="page.currPage" value="page.currPage+1"></s:param>
            </s:a>
</s:if><s:else>
            下一页
                </s:else> </span> <span> <s:if test="page.hasNext==true">
                <s:a action="%{pageAction}">
                最后一页
                    <s:param name="page.index"
                        value="(page.pageCount-1)*page.pageSize"></s:param>
                    <s:param name="page.currPage" value="page.pageCount"></s:param>
                </s:a>
</s:if><s:else>
                最后一页
                    </s:else> </span>
</div>
```

分页后台是一个普通 Java 类，主要提供一些分页重用的方法，如记录总数、当前页数和当前索引数等，关键代码实现如下。

```
public class PageUtil {
    private Integer pageSize = 10;           // 一页显示条数，默认为 10
    private Integer recordCount = 0;         // 总条数
    private Integer index = 0;               // 索引下标
    private Integer currPage = 1;           // 当前页数
```

```
..... // 省略了 setter 和 getter 方法  
}
```

使用分页只需要把分页的 Bean 注入到 Action 中，并且把分页组件的 JSP include 到目标 JSP 页面即可使用。

```
<!-- 为分页定制的 url，支持传参数 -->  
<s:url id="pageAction" includeContext="false"  
action="articleAction_queryAllMyArticles" namespace="/">  
</s:url>  
<!-- 分页 -->  
<s:include value="/WEB-INF/jsp/pageUtil.jsp"></s:include>
```

📢 注意：在调用分页的组件时 URL 的 id 一定要和分页组件中的 Action 属性匹配；另外，传入 URL 时也可以传递参数。

B.7 主页面设计

明日知道系统的主页面可分为两大类，分别为文章搜索的首页和论坛的首页。下面分别进行介绍。

B.7.1 文章搜索首页设计

在文章搜索首页中用户可以搜索出相关的文章，并且在该页面中还包括“登录”、“注册”、“进入论坛”超链接，运行结果如图 B.9 所示。



图 B.9 首页运行结果

文章的每种类型都是一张图片，在页面加载时动态创建。动态加载图片的代码如下。

```
var articleTypes = {
  'Java': // 名称
  {
    id: 'Java', // id
    style: 'cursor:hand;', // 样式,
    src: 'images/top_02.gif', // 图片
    activeSrc: 'images/top2_02.gif', // 被激活时的图片
    width: 98,
    height: 35
  },
  ..... // 其他文章类型省略不写
}
```

在首页中通过 jQuery 框架加载文章类型，具体代码如下：

```
/**
 * 加载文章类型 title
 */
var activeId = ""; //选中的文章类型
$(function() {
  var div = $('#articleTypeDiv');
  for ( var type in articleTypes) {
    var articleType = articleTypes[type];
    var img = $('<img>');
```

```
img.attr('src', articleType.src);
img.attr('activeSrc', articleType.activeSrc);
img.attr('height', articleType.height);
img.attr('width', articleType.width);
img.attr('id', articleType.id);
img.attr('style', articleType.style);
img.attr('border', "0");
img.attr('alt', "");
img.bind('mouseover', function() {
var o = $(this);
    if (o.attr('id') != activeId) {
        o.attr('src', articleTypes[o.attr('id')].activeSrc);
    }
});
img.bind('click', function() {
var o = $(this);
    o.attr('src', articleTypes[o.attr('id')].activeSrc);
    if (activeId != "") {
        if (o.attr('id') == activeId) {
            o.attr('src', articleTypes[o.attr('id')].src);
            activeId = "";
            return;
        } else {
            document.getElementById(activeId).src =
articleTypes[activeId].src;
        }
    }
    activeId = o.attr('id');
});
img.bind('mouseout', function() {
var o = $(this);
    if (o.attr('id') != activeId) {
        o.attr('src', articleTypes[o.attr('id')].src);
    }
});
div.append(img);
}
doSearchForm.reset();
});
```

B.7.2 论坛页设计

论坛页显示所有的文章类型、类型描述、文章和回复次数，以及文章的动态信息，单击某个文章类型，可以搜索其下的所有文章；单击文章作者，可以搜索其发表的所有文章。论坛首页如图 B.10 所示。

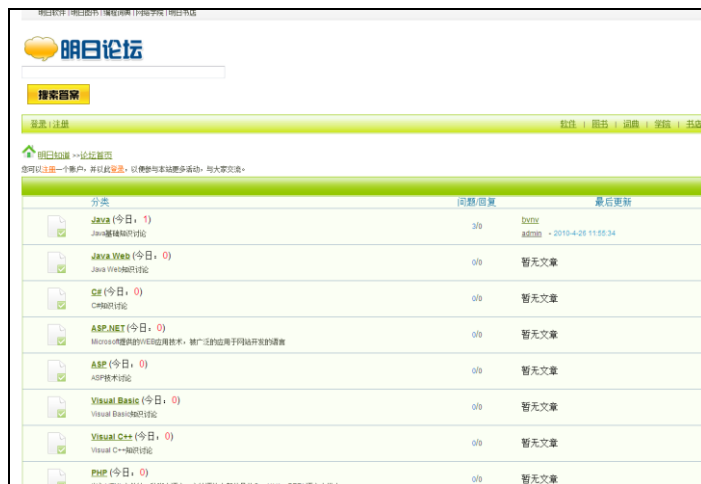


图 B.10 论坛首页

在论坛首页中，用户可以浏览到每个类型文章的问题数与浏览数，最后更新等信息，论坛首页代码是用 Struts2 标签来完成，部分关键代码如下：

```
<s:iterator value="articleTypes" id="articleType" status="st">
<s:a cssClass="hong" ction="articleAction_findArticlesByType" target="_blank">
<s:property value="#articleType.articleTypeName" />
<s:param name="articleType"
    value="#articleType.articleTypeName">
</s:param>
</s:a>
.....
// 其他代码省略
</s:iterator>
```

 **说明：**articleType 参数：该参数为一个 List<ArticleType> articleType 可以看做是 articleType 的一个对象，通过它可以获取 Article 实体类中的属性值。

在论坛首页中显示了所有文章类型，本系统中在 Action 中调用 DAO 中查询所有文章类型方法，Dao 中定义查找文章类型方法代码如下：

```
/**
 * 查询所有文章类型
 *
 * @return
 */
public List<ArticleType> queryAllArticleType() {
    return this.getCriteria(ArticleType.class).list();
}
```


在 ArticleAction 中定义调用 Dao 类中方法，获取查询结果，并将请求转发至首页。具体代码如下：

```
/**
 * 进入论坛首页
 *
 * @return
 */
public String forum() {
```

```

        articleTypes = articleDao.queryAllArticleType();
        return "forum";
    }

```

 **说明：**本系统中的所有 DAO 层类均继承了 DefaultDaoImpl 类，这样即可直接调用已经封装的持久层方法。

B.8 文章维护模块设计

文章维护功能模块主要包括文章的添加、修改、删除、浏览及回复等子功能模块。

B.8.1 添加文章模块

已登录的用户进入论坛首页，单击“添加文章”超链接打开“添加文章”页面即可添加文章，如图 B.11 所示。



图 B.11 添加文章

用户输入信息后系统会验证输入内容的合法性，以防止非法的数据破坏系统，如非法字符和超长文字等。添加文章表单和验证代码如下。

```

<form action="articleAction_addArticle" method="post"
id="addArticleForm">
<table>
<tr>
<td class="huise">文章标题: </td>
<td><input type="text" name="article.title" id="title"></td>
</tr>
<tr>
<td class="huise">所属类型: </td>
<td>
<select name="article.articleTypeName" id="type">
<option value="">请选择</option>
<option value="Visual Basic">Visual Basic</option>

```

```

        <option value="Visual C++">Visual C++</option>
        <option value="Java">Java</option>
        <option value="Java web">Java web</option>
        <option value="C#">C#</option>
        <option value="ASP.NET">ASP.NET</option>
        <option value="PHP">PHP</option>
        <option value="ASP">ASP</option>
        <option value="其他">其他</option>
    </select>
</td>
</tr>
<tr>
    <td class="huise">文章内容: </td>
    <td><textarea name="article.content" cols="80"
rows="10" id="content"></textarea></td>
</tr>
</table>
<p align="center"><input type="button" value="发表文章"
onclick="addArticle1()" /></p>
</form>

```

单击“发表文章”按钮，系统校验输入是否为空，通过 JavaScript 代码实现，具体代码如下：

```

<SCRIPT type="text/javascript">
function addArticle1() {
if (!$('#title').val()) {
alert('请输入标题');
return;
}
if (!$('#type').val()) {
alert('请选择文章类型');
return;
}
if (!$('#content').val()) {
alert('请输入文章内容');
return;
}
addArticleForm.submit();
}
</SCRIPT>

```

页面将参数传给 Action，Action 将参数封装为文章对象传递给 DAO 层，DAO 层调用保存方法即可把文章信息存入数据库。文章对象类的代码如下。

```

public class Article {
    private Integer articleId = null;        // 文章主键 id
    private String title = null;             // 标题
    private String content = null;           // 内容
    private Date emitTime = null;            // 发表时间
    private Date lastUpdateTime = null;      // 最后更新时间
    private String articleTypeName = null;   // 文章类型名称
    private User user = null;               // 文章作者
    private ArticleType articleType = null;  // 文章类型
    private Set<Reply> replies = null;       // 文章回复
    private Set<Scan> scans = null;          // 文章浏览
    .....                                   // 省略 getter 和 setter 方法
}

```


在 `ArticleDaoImpl` 类中调用添加文章方法 `addArticle()`, 该方法有一个 `Article` 类型参数, 用于表示要添加的文章类型。通过 `Spring` 框架实现文件添加操作很简单, 具体代码如下:

```
/**
 * 添加文章
 */
public void addArticle(Article article) {
    ArticleType articleType = this.getArticleTypeByName(article.getArticleTypeName());
    article.setArticleType(articleType);
    this.save(article);
}
```

B.8.2 浏览文章

单击“查看详细”超链接或者文章标题即可打开文章的详细信息页面, 如图 B.12 所示。

当用户单击“进入文章”超链接后页面向后台传一个文章 `id`, 系统根据这个 `id` 通过持久层查询单篇文章的方法即可获取该篇文章的所有信息。然后将文章对象传回 `Struts 2`, `Struts 2` 根据文章属性信息显示文章信息。

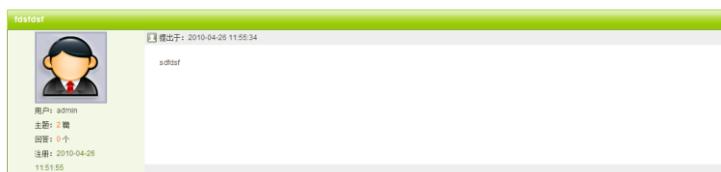


图 B.12 文章的详细信息页面

在浏览文章页面中, 通过 `Struts2` 标签, 将查询出来的文章信息显示在页面中, 具体代码如下:

```
<tr>
<td width="160" class="huise1">用户:
<s:property value="article.user.userName" /><br />
主题: <span class="chengse">
<s:property value="article.user.myArticleCount" />
</span> 篇<br />
回答: <span class="chengse" id="replyCount">
<s:property value="article.user.myReplyCount" />
</span> 个<br />
注册: <span
class="henhong"><s:date name="article.user.registerTime"
format="yyyy-MM-dd hh:mm:ss" /></span></td>
</tr>
..... // 其他代码省略
<tr>
<td width="24" align="center"></td>
<td width="173">提出于: <s:date name="article.emitTime"
format="yyyy-MM-dd hh:mm:ss" /></td>
</tr>
```

```
..... // 其他代码省略
<div style="width: 50"><s:property value="article.content" /></div>
```

在 `ArticleDaoImpl` 类中，定义按照文章编号查询文章信息方法 `querySingleArticle()`，该方法有一个 `String` 类型的参数，用于指定要查询的文章编号，关键代码如下：

```
/**
 * 查找单篇文章
 */
public Article querySingleArticle(String articleId) {
    String hql = "from Article where articleId=" + articleId;
    return (Article) this.find(hql).get(0);
}
```

B.8.3 文章回复

用户浏览文章之后可以对文章进行回复，但前提是用户必须已经登录系统，否则不能对文章进行回复。文章回复效果如图 B.13 所示。

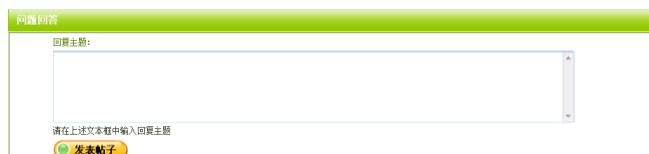


图 B.13 回复文章页面

本系统应用了 `Hibernate` 开发框架，首先编写与文章回复表对应的持久化类 `Reply`，该类中包含的属性与文章回复表中的字段一一对应，具体代码如下：

```
public class Reply {
    private Integer replyId = null; // 回复主键 id
    private Date replyTime = null; // 回复时间
    private String content = null; // 回复内容
    private User user = null; // 回复用户
    private Article article = null; // 回复的文章
    ..... //getter 和 setter 方法省略
}
```

在 `ReplyDaoImpl` 类中，编写保存文章回复方法 `addReply`，该方法有一个持久化类 `Reply` 类型对象，调用 `Hibernate` 的 `save()` 方法，实现保存操作，具体代码如下：

```
/**
 * 添加回复
 */
public void addReply(Reply reply) {
    this.save(reply);
}
```

B.8.4 修改文章

用户从“我的文章列表”中选择一篇文章，系统判断该文章的作者是否当前用户。如果是，则显示“修改”按钮，用户才有权修改文章。“修改文章”页面如图 B.14 所示。

修改文章首先根据文章 id 查询文章，并为需要修改的属性赋值，然后执行 `update`。修改文章的 Action 代码如下。

图 B.14 “修改文章”页面

```
/**
 * 修改文章
 * @return
 */
public String updateArticle() {
    Article article = articleDao.querySingleArticle(this.article.getId().toString());
    article.setLastUpdateTime(new Date());
    article.setTitle(this.article.getTitle());
    article.setContent(this.article.getContent());
    this.articleDao.updateArticle(article);
    this.article = articleDao.querySingleArticle(this.article.getId().toString());
    return "singleArticle";
}
```

B.8.5 删除文章

与修改文章一样，用户只能删除自己的文章。在页面中通过 Struts2 标签判断页面是否显示“删除”按钮，代码如下：


```
<s:a action="articleAction_deleteArticle" cssClass="hong">
<s:param name="article.articleId"
value="article.articleId"></s:param>
删除
</s:a>
```

删除文章之后，再做一次查询，页面将跳转到我的文章列表。具体代码如下：

```
/**
 * 删除文章
 * @return
 */
public String deleteArticle() {
    articleDao.deleteArticle(this.article); //删除所选文章
    User user = new User();
    user.setUserId(this.getCurrentUser().getUserId()); //设置用户信息
    this.article.setUser(user);
    //根据用户信息，查询其发表的所有文章
    this.myArticles = this.articleDao.queryAllArticleByUser(user, this.getFirstResult(), this.getMaxResults());
    return "myArticle";
}
```

删除文章的时候，需要删除该文章下的所有的回复信息以及浏览信息，否则数据库将会产生冗余数据。为了达到级联删除，只需要在 Hibernate 映射文件配置就可以了。

```
<set name="replies" inverse="true" cascade="all" order-by="replyTime desc">
<key column="articleId" />
<one-to-many class="Reply" />
</set>
<set name="scans" inverse="true" cascade="all" order-by="scanTime desc">
<key column="articleId" />
<one-to-many class="Scan" />
</set>
```

 说明：cascade="true" 为级联删除，order-by="replyTime desc" 表示以时间倒序来排序，<key column="articleId" /> articleId 为关联外键。

B.9 文章搜索模块设计

文章搜索模块是本系统的核心模块，其中包括多个功能。

B.9.1 搜索我的文章

用户登录论坛之后，单击“我的文章”超链接搜索用户发表过的所有文章，“我的文章列表”页面如图 B.15 所示。



图 B.15 “我的文章列表”页面

搜索我的文章主要流程是系统先取得当前用户信息，传给后台根据文章对象里的用户 id，查询出该用户下的所有文章。具体代码实现如下：

```
/**
 * 查询自己发表的所有文章
 * @return
 */
public String queryAllMyArticles() {
    this.myArticles = this.articleDao.queryAllArticleByUser(this.getCurrUser(),
        this.getPage().getIndex().toString(), this.getPage().getPageSize().toString());
    this.getPage().setRecordCount(this.articleDao.queryAllArticleCountByUser(this.getCurrUser()));
    return "myArticle";
}
```


定义方法 queryAllArticleByUser()，用于查找某个用户发表的所有文章。在实现查询的过程中，使用了分页技术。具体代码如下：

```
/**
```

```

* 查找某个用户发表的所有文章
*/
public List<Article> queryAllArticleByUser(User user, String firstResult, String maxResults) {
    String hql = "from Article where userId=" + user.getUserId() + "order by emitTime desc";
    return this.query(hql, firstResult, maxResults);
}

```

 说明：query()方法的3个参数分别代表HQL语句、分页查询的索引值和一页的数据记录数，其实现在DefaultDaoImpl中。

B.9.2 根据关键字搜索文章

根据文章关键字搜索文章时系统使用输入关键字中文章的标题和内容匹配任何位置，如果匹配成功，则返回搜索结果；否则返回空。3种情况是在明日论坛首页输入关键字、在明日论坛首页选择一个文章类型再加上关键字和进入论坛输入关键字。“符合条件的文章列表”页面如图B.16所示。



图 B.16 “符合条件的文章列表”页面

根据关键字搜索文章的前台代码如下。

```

<table width="480" border="0" align="center" cellpadding="0"
cellspacing="0">
<s:hidden name="article.articleTypeName"
id="articleTypeName"></s:hidden>
<tr>
<td width="378" height="35">
<table width="359" height="35" border="0" cellpadding="0"
cellspacing="0">
<tr>
<td align="center">
<input type="text" id="searchStr"
name="searchStr" style="width: 350px; height:
20px;" />
</td>
</tr>
</table>
</td>
<td width="113">

</td>
</tr>
</table>

```

当用户在系统首页，或论坛首页中单击“搜索”按钮，系统会执行的 JavaScript 方法，判断用户输入的搜索内容是否合法。JavaScript 具体代码如下：

```
/**
 * 搜索文章
 * @return
 */
function doSearch() {
    var searchText = $.trim($('#searchStr').val());
    if (!searchText) {
        alert('请输入要搜索的内容');
        return;
    }
    if (searchText.length > 255) {
        alert('输入内容不能超过 255 个字符');
        return;
    }
    $('#articleTypeName').val(activeId);
    doSearchForm.submit();
}
```

在 ActicleAction 中，定义 doSearch()方法实现通过关键字搜索文章，并将请求转发至相应地址，具体代码如下：

```
/**
 * 通过关键字搜索文章
 * @return
 */
public String doSearch() {
    if (searchStr != null) {
        searchStr = searchStr.trim();
    }
    String type = this.article == null ? null : this.article.getArticleTypeName();
    this.searchArticles = this.articleDao.doSearch(type, searchStr, this.getFirstResult(), this.
    getMaxResults());
    return "searchResult";
}
```


在 ArticleDaoImpl 中定义根据用户输入内容搜索符合条件的文章方法 doSearch。该方法有 4 个 String 类型的参数，分别用于指定要搜索的文章类型，要搜索的文章内容，分页显示的参数等。将查询结果以 List 形式返回。该方法采用的是 QBC 方式进行查询，这种方式的优点是不用手动写 HQL 语句，也不用考虑一些 SQL 关键字的注入攻击（例如%、*、[]等特殊字符），只需要简单调用 Criteria 提供的简单方法就行。具体代码如下：

```
/**
 * 根据输入内容搜索符合条件的文章
 */
@SuppressWarnings("unchecked")
```

```

public List<Article> doSearch(String type, String str, String firstResult,
    String maxResults) {
    int first = new Integer(firstResult).intValue();
    int max = new Integer(maxResults).intValue();
    Criteria criteria = this.getCriteria(Article.class);
    if (type != null && !type.equals("")) {
        criteria.add(Restrictions.eq("articleTypeName", type));
    }
    criteria.add(
        Restrictions.or(Restrictions.like("title", str,
            MatchMode.ANYWHERE), Restrictions.like("content", str,
            MatchMode.ANYWHERE))).addOrder(
        Order.desc("lastUpdateTime")).setFirstResult(first)
        .setMaxResults(max);
    List<Article> list = criteria.list();
    return list;
}

```

 **说明：**Criteria 对象可以添加多个表达式，在该实例中添加的表单式包括根据关键字在任意位置进行匹配、按照文章的最后更新时间进行倒序排序和分页表达式。Criteria 使得开发人员可以写更少的代码去完成更多的功能。

B.9.3 热门搜索

每个热门搜索都是一个超链接，用户只要进入一个热门搜索，系统即可查询出有关该热门的所有文章，如图 B.17 所示。



图 B.17 热门搜索

热门搜索的前台代码如下。

```

<table width="480" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td><span class="danhuang02">明日论坛热门搜索: </span>
    <spanclass="cubai"><a href="#" onclick="seatchHot('c#')">C#</a>
    &nbsp; &nbsp; &nbsp; <a href="#" onclick="seatchHot('Java 编程词典')">Java 编程词典</a>
    &nbsp; &nbsp; &nbsp; <a href="#" onclick="seatchHot('Java Web')">Java Web</a> &nbsp; &nbsp; &nbsp;
    <a href="#" onclick="seatchHot('Java 从入门到精通')">Java 从入门到精通</a> &nbsp; &nbsp; </span></td>
  </tr>
</table>

```

单击“搜索”按钮，系统会将热门搜索的字符串赋给搜索框的值，这样执行逻辑与普通搜索相同，JavaScript 赋值代码如下。

```

/**
 * 热门搜索
 * @param content
 * @return
 */
function seatchHot(content) {
    $('#searchStr').val(content);
}

```

```
doSearchForm.submit();
}
```

B.9.4 搜索文章作者的所有文章

当单击文章作者，系统会搜索该作者发表过的所有的文章。单击文章类型打开的页面如图 B.18 所示。

| 分类 | 问题/回复 | 最后更新 |
|--|-------|-------------------------------------|
|  Java (今日: 1) Java 基础知识讨论 | 3/0 | gdu yiaolin - 2010-11-1 11:17:06 |
|  Java Web (今日: 0) Java Web 知识讨论 | 0/0 | 暂无文章 |
|  C# (今日: 0) C# 知识讨论 | 0/0 | 暂无文章 |
|  ASP.NET (今日: 0) Microsoft 提供的 WEB 应用技术，被广泛地应用于网站开发的语言 | 0/0 | 暂无文章 |
|  ASP (今日: 0) ASP 技术讨论 | 0/0 | 暂无文章 |
|  Visual Basic (今日: 0) Visual Basic 知识讨论 | 0/0 | 暂无文章 |
|  Visual C++ (今日: 0) Visual C++ 知识讨论 | 0/0 | 暂无文章 |
|  PHP (今日: 0) 嵌入 HTML 文件的一种脚本语言。它的语法大部分是从 C、JAVA、PERL 语言中借来 | 0/0 | 暂无文章 |

图 B.18 单击文章类型打开的页面

按照作者查询文章页面的关键代码如下所示。

```
<s:a action="articleAction_queryArticlesByUserOfArticle" cssClass="huise">
  <s:property value="#article.user.userName" />
  <s:param name="article.articleId" value="#article.articleId"></s:param>
  <s:param name="user.userName" value="#article.user.userName"></s:param>
</s:a>
```

在 `ArticleAction` 中定义 `queryArticlesByUserOfArticle()` 方法，实现查询文章作者的所有文章。并将请求转发至相应地址。具体代码如下：

```
/**
 * 查询文章作者的所有文章
 * @return
 */
public String queryArticlesByUserOfArticle() {
    this.searchArticles = this.articleDao.findArticlesByUserOfArticle(
        this.article.getId().toString(), this.firstResult, this.maxResults);
    return "userArticle";
}
```

在 `ArticleDaoImpl` 类中定义方法，二级缓存中取出用户信息，再根据该用户信息，查询出其发表的所有文章。代码如下：

```
/**
 * 查找文章发表人发表过的所有文章
 * @param articleId
```



```

* @param firstResult
* @param maxResults
* @return
*/
public List<Article> findArticlesByUserOfArticle(String articleId,String first Result, String maxResults) {
    Article article = this.querySingleArticle(articleId);
    User user = article.getUser();
    return queryAllArticleByUser(user, firstResult, maxResults);
}
/**
* 查找某个用户发表的所有文章
*/
public List<Article> queryAllArticleByUser(User user, String firstResult, String maxResults) {
    String hql = "from Article where userId=" + user.getUserId()+ "order by emitTime desc";
    return this.query(hql, firstResult, maxResults);
}

```

B.9.5 搜索回复作者的所有文章

在 ActicleAction 类中定义 queryArticlesByUserOfReply()方法, 实现查询文章回复用户的所有文章。并将请求转发至相应地址, 具体代码如下:

```

/ **
* 查询文章回复用户的所有文章
* @return
*/
public String queryArticlesByUserOfReply() {
    this.searchArticles = this.articleDao.findArticlesByUserOfReply(this. reply.getReplyId(). toString(),
    this.firstResult,this.maxResults);
    return "userArticle";
}

```

在 ArticleDaoImpl 类中定义 findArticlesByUserOfReply()方法, 实现查询回复人发表过的所有文章。具体代码如下:

```

/ **
* 查找回复人发表过的所有文章
* @param replyId
* @param firstResult
* @param maxResults
* @return
*/
public List<Article> findArticlesByUserOfReply(String replyId,String first Result, String maxResults) {
    Reply reply = (Reply) this.load(Reply.class, replyId);
    User user = reply.getUser();
    return queryAllArticleByUser(user, firstResult, maxResults);
}

```

B.10 运 行 项 目

项目开发完成后，就可以在 Eclipse 中运行该项目了，具体步骤参见 5.6 节。

B.11 本 章 小 结

本章为大家介绍了 3 大流行框架整合开发的明日论坛，本系统中除了应用 3 大框架外，还应用了 jQuery 进行前台开发，所用技术都是当前最流行的，也是读者最感兴趣的，认识阅读本章，并仔细研究项目的读者，相信会对本系统应用的技术有很大的提高。