博客网站系统分析与设计

编制时间：2016.5.1

版本号：V1.0

文档变更记录：

日期	修订号	描述	编制者	审阅者

目录

第 1 章 引言

1.1 文档用途

编写本文档的主要目的是用于说明博客网站系统的需求分析和各个用例的具体实现。

1.2 阅读对象

本项目组的成员和所有对该博客网站系统感兴趣的人。

1.3 系统适用对象

本系统适合所有想用文字记录生活、分享经验知识、表达自我的人。思维通只有通过文字的梳理才能变得清晰明澈，思想只有通过彼此的交流摩擦才能激发出智慧的火花，通过文字的方式让分享成为一种习惯和精神。所以，所有想通过与他人分享、交流来提高自己知识、见解、能力的人都可以选择使用本系统。

第二章 问题陈述

2.1 发现问题？

在互联网飞速发展的今天，人们的大脑已经被碎片化信息的洪流淹没，我们急于在没有营养的网络快餐文化中汲取养分，最终却只能营养不良。我们已经忘记上一次静下心来好好写作是什么时候了，我们的心太浮躁了，我们需要找点时间，静下心来好好梳理一下自己的思维。虽说现在以博客网站为主流的写作平台已经发展得相当成熟，拥有庞大的用户基数，但作为商业化的产品，它们就势必要考虑一些利益相关的问题，但一谈到利益就势必会产生一些不愉快的事情，反反复复弹来弹去的广告推广影响了我们的阅读情绪，有特定运营目的潮流导向影响了我们写作的初衷，我们本想安安静静地和彼此交流、分享思想见解和经验知识，但却又被带入了一个庸俗、聒噪的环境。

2.2 解决办法？

基于 2.1 中提出的问题，我们打算设计一个纯粹的写作平台，没有那些多余的华而不实的装饰，也没有纷繁复杂的偏社交导向的互动功能，我们只设计几

个简单的功能：写日志、阅读日志、评论日志，甚至都不设置点赞和收藏功能，对于一个虔诚的写作者和读者，那些细微的虚荣都是多余的。

2.3 如何实现？

由此我们设计了这个博客网站系统，专门为那些想寻一片静地好好阅读和写作的人提供一个干净、纯粹的写作平台。

首先，为了吸引用户量，用户在未注册登录之前便可阅读博客首页他人的博文。

用户注册登录之后，可在“个性化设置”栏目中设置自己的博客标题和个性化签名。可以在“写日志”栏目编辑新的日志，或者在用户首页查看自己的过往日记。

第三章 需求分析

3.1 博客网站系统用例图

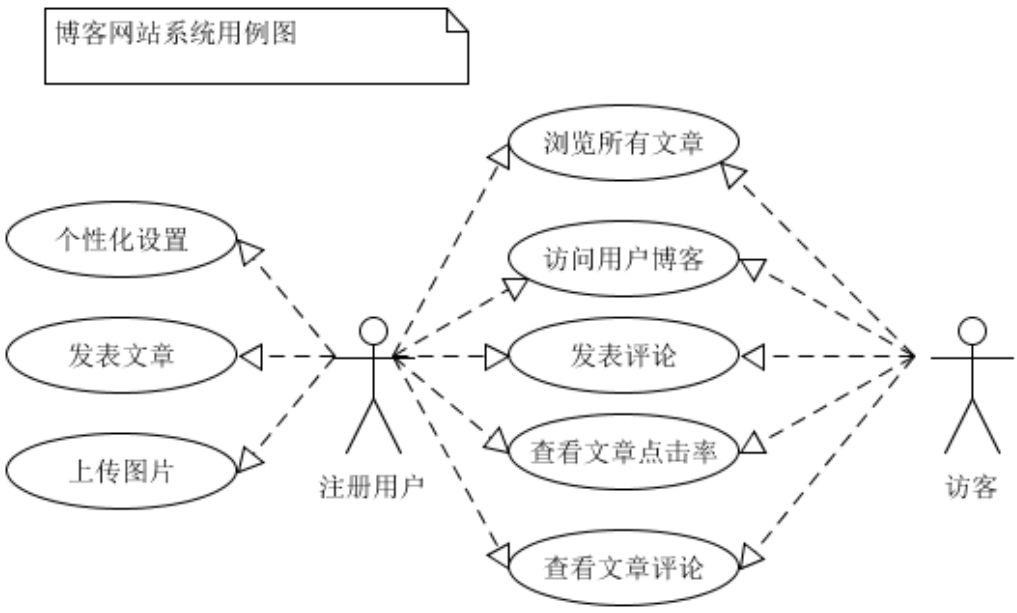

图 1 博客网站系统用例图

3.2 博客网站系统用例规约

3.2.1 用户登录

用例 ID	UC201
用例名	用户登录
简要描述	本用例描述注册用户如何登录到博客网站系统。
参与者	注册用户
前置条件	本用例开始前，注册用户已经打开系统登录界面。
基本事件流	本用例开始于注册用户希望登录到博客网站系统。 1.系统请求注册用户输入用户名和密码。 2.注册用户输入用户名和密码。 3.系统验证输入的用户名和密码。 A1.用户名不存在。 A2.用户名对应的密码不正确。 4.注册用户成功登录到主界面，进行其他操作。
备选事件流	A1.用户名不存在： ➢ 系统提示用户名不存在错误。 ➢ 系统提示用户进行注册。 A2.用户名对应的密码不正确： ➢ 系统提示用户密码不正确错误。 ➢ 返回基本事件流第 1 步。

活动图

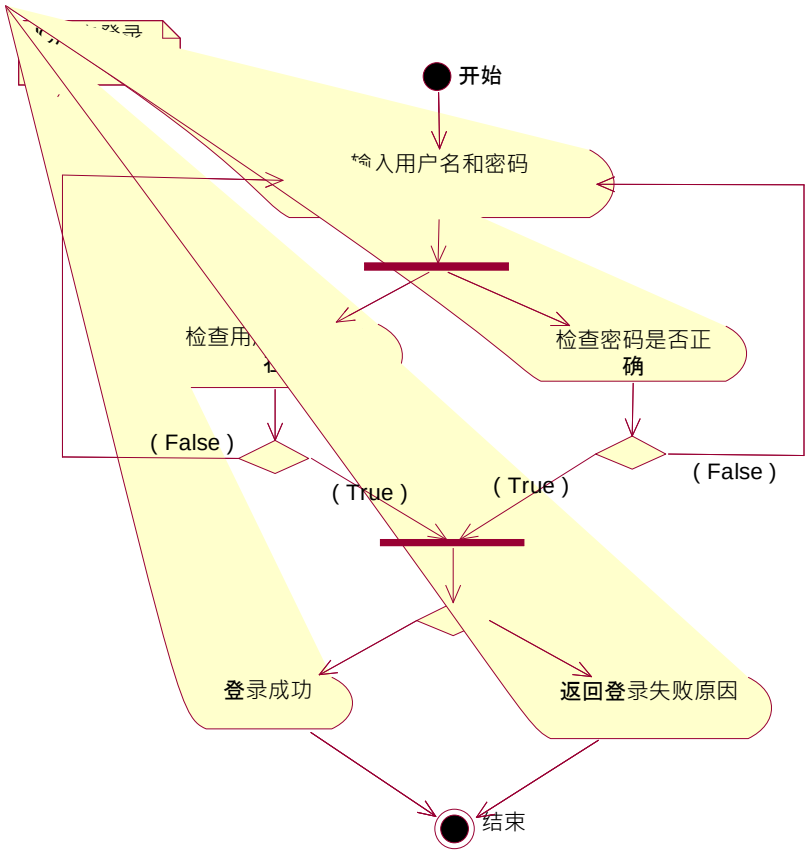

图 2 用户登录活动图

后置条件　若用例成功，注册用户将成功登录系统。若失败，系统状态不改变。

特殊需求　1. 密码输入框必须以密文方式呈现。

2. 如果是密码错误，在连续输入 5 次之后，不能再继续输入密码，用户只能通过找回密码功能找回密码。

补充说明　当用户名不存在时，系统提示用户注册，但不直接转到注册界面，用户可以选择再次输入用户名，实在想不起来再选择重新注册。(不提供找回用户名功能)

3.2.2 用户注册

用例 ID　UC202

用例名　用户注册

简要描述	本用例描述新用户如何在博客网站系统中进行注册。 1.用户名合法性定义：用户名只能包含字母、数字以及下划线且长度超过 6 个字符少于 20 个字符。 2.密码合法性定义：密码的长度必须超过六 6 个字符。
参与者	注册用户
前置条件	本用例开始前，注册用户已经打开系统注册界面。
基本事件流	本用例开始于注册用户希望在博客网站系统中注册。 1.系统请求注册用户输入用户名： A1.用户名已经存在。 A2.用户名不合法。 2.系统请求注册用户两次输入密码： A3.密码不合法。 A4.两次密码输入不一样。 3.系统请求用户输入姓名，性别，电子邮件。 4.系统把当前用户的信息增加到数据库中。
备选事件流	A1.用户名已经存在： ➢ 系统发出“用户名已存在”错误提示。 ➢ 返回基本事件流第 1 步。 A2.用户名不合法： ➢ 系统发出“用户名不合法”错误提示。 ➢ 返回基本事件流第 1 步。 A3.密码不合法： ➢ 系统发出“密码不合法”错误提示。 ➢ 返回基本事件流第 2 步。 A4.两次密码输入不一样： ➢ 系统发出“两次密码输入不一样”错误提示。 ➢ 返回基本事件流第 2 步。

活动图

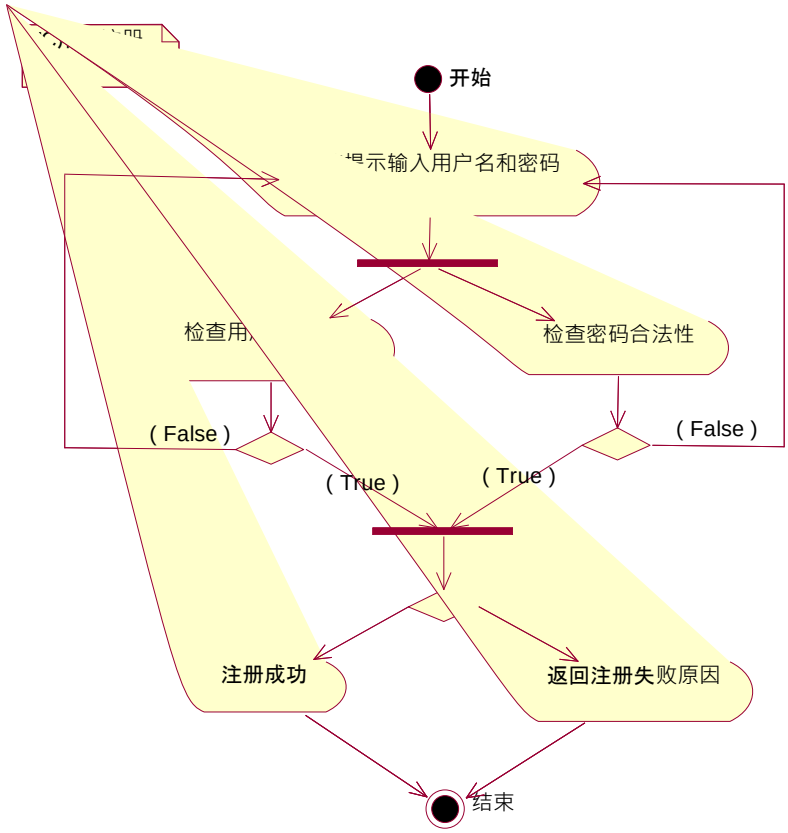

图 3 用户注册活动图

后置条件　　如果用例成功，用户将注册系统成功，系统在数据中增加一条注册用户的相关记录。若失败，系统状态不改变。

特殊需求　　1.密码输入框必须以密文方式呈现。

2.为了避免密码碰撞以及字典攻击，应考虑采用盐值加密的方法将密码保存到数据库中。

补充说明　　支持第三方注册登录。

3.2.3 个性化设置

用例 ID　　UC203

用例名　　个性化设置

简要描述　　本用例允许注册用户对博客标题和个性签名进行重新编辑。

参与者　　注册用户

前置条件　本用例开始前注册用户已经登录系统。

基本事件流　本用例开始于注册用户想修改博客标题和个性签名。

1. 系统请求注册用户输入博客标题和个性签名。

2. 系统修改注册用户的博客标题和个性签名信息并在页面相应位置展示。

备选事件流　A1. 输入字符不合法：

- 返回基本事件流第 1 步。

活动图

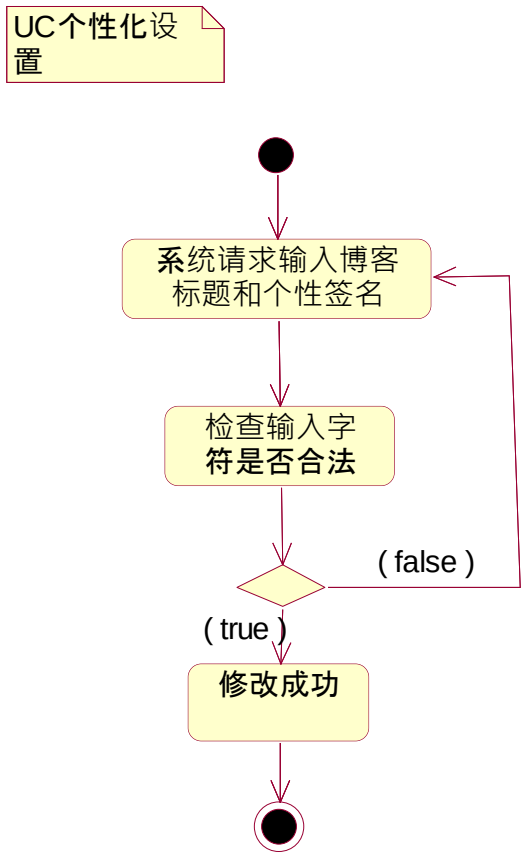

图 4 个性化设置活动图

后置条件　如果用例成功，注册用户将成功修改博客标题和个性签名。

特殊需求　输入字符只能是常规的汉字或者英文字符。

补充说明　如果修改不成功，默认不进行任何处理。

3.2.4 写日志

用例 ID	UC204
用例名	写日志
简要描述	本用例允许注册用户在登录后编辑发表自己的日志。
参与者	注册用户
前置条件	本用例开始前注册用户已经登录系统。
基本事件流	本用例开始于注册用户想编辑发表日志。 1.注册用户选择写日志。 2.系统提示登录用户编辑标题和正文内容。 3.编辑好后，用户提交日志。 A1.用户没有提交就离开该页面。 4.如果提交成功，系统跳转到我的全部文章页面。
备选事件流	A1.用户没有提交就离开该页面。 ➢ 系统跳转到目标页面，原来页面的数据不会被保存。
活动图	

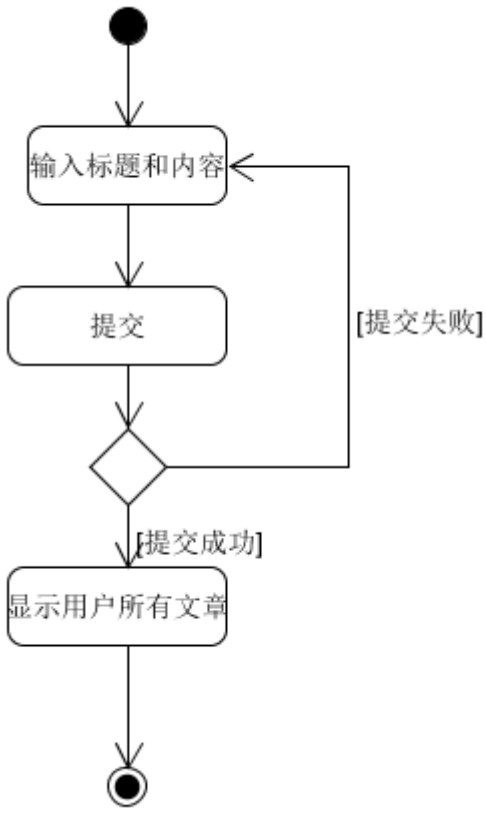

图 5 写日志活动图

后置条件	如果用例成功，注册用户将成功发布日志。若失败，系统状态不改变。
特殊需求	无。
补充说明	如果提交不成功，默认不做任何处理。

3.2.5 传图片

用例 ID	UC205
用例名	传图片
简要描述	本用例允许注册用户上传照片到博客网站。
参与者	注册用户
前置条件	本用例开始前用户已经登录系统。
基本事件流	本用例开始于注册用户想要上传照片到博客网站系统。 1.注册用户选择进入相册。 2.注册用户选择文件。 3.注册用户在本地系统选择好照片后，选择上传。 A1.用户没有上传直接退出。 4.如果上传成功，系统显示相册图片。
备选事件流	A1.用户没有上传直接退出： ➢ 系统返回第 2 步。

活动图

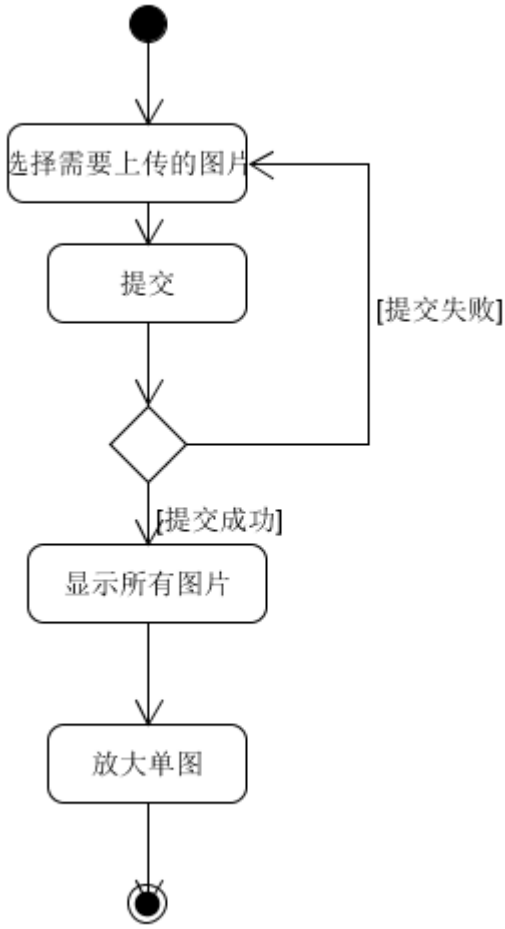

图 6 传图片活动图

后置条件	如果用例成功，用户将成功上传图片。若失败，系统状态不改变。
特殊需求	无。
补充说明	如果上传不成功，默认不做任何处理。

3.2.6 阅读

用例 ID	UC206
用例名	阅读
简要描述	本用例允许所有访问系统的用户阅读自己或他人的博客，包括日志和照片。

参与者　所有访问系统的用户

前置条件　本用例开始前所有访问系统的用户已经打开系统，并不要求登录。

基本事件流　本用例开始于所有访问系统的用户想要阅读自己和他人的博客，包括日志和照片。

1.所有访问系统的用户选择博客首页。

2.点击博客标题进入阅读。

活动图

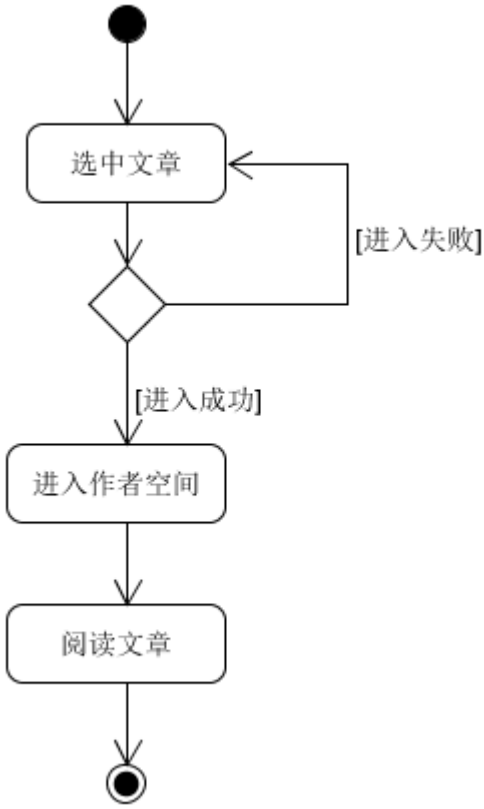

图 7 阅读活动图

后置条件　如果用例成功，所有访问系统的用户都能阅读自己或他人的博客，包括日志和照片。若失败，系统状态不改变。

特殊需求　获取文章或图片的响应速度要快。

补充说明　无。

3.2.7 评论

用例 ID	UC207
用例名	评论
简要描述	本用例允许注册用户评论正在阅读的博文。
参与者	注册用户
前置条件	本用例开始前用户已经成功登录系统。
基本事件流	本用例开始于注册用户想要评论正在阅读的博文。 1.系统请求注册用户输入评论。 2.注册用户编辑评论内容。 3.注册用户提交评论。 A1.用户没有提交直接退出。
备选事件流	A1.用户没有提交直接退出： ➢ 返回主流程第 1 步。

活动图

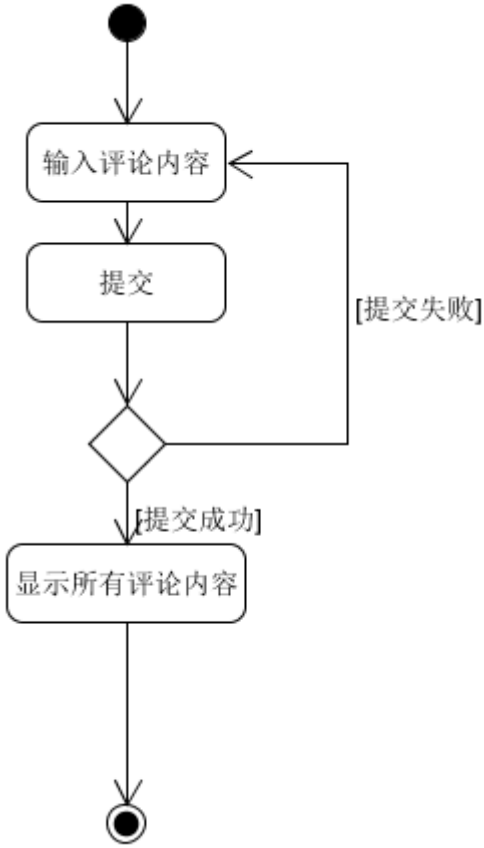

图 8 评论活动图

后置条件	如果用例成功，注册用户评价博文成功。若失败，系统状态不改变。
特殊需求	无。
补充说明	无。

3.3 补充规约

3.3.1 目标

本补充规约用于描述那些在用例模型中没有描述到的非功能性需求，大多是全局性非功能需求，补充规约和用例模型共同定义了系统的全部需求。

3.3.2 范围

本补充规约主要描述了系统的非功能性需求，包括：可用性，可靠性，易用性，安全性等。

3.3.3 可用性

因为系统在工作时需要不断地从服务器数据库中取出数据和输入数据，所以系统要在正常联网状态下才能正常使用。

3.3.4 可靠性

系统要保证用户数据读取和输入保存无误，新建日志，发布日志，发布评论等操作能正常进行。

3.3.5 易用性

功能导航设置合理，功能设计主次分明，页面的交互逻辑符合大多数人的习惯。

3.3.6 安全性

用户密码用 MD5 加密，防止密码泄露，用户数据根据不同的权限设置进行不同级别的加密操作。

3.4 术语表

表 1 术语表

术语	英语名称	定义和信息	别名
博客网站系统	Blog Website System	一个沉浸式的写作平台，用户可以在上面阅读和发表文章并能对文章进行评论。	
日志	Blog	这里主要指“网络日志”，也即 blog,可以理解为网上日记。	

3.5 绘制系统流程图

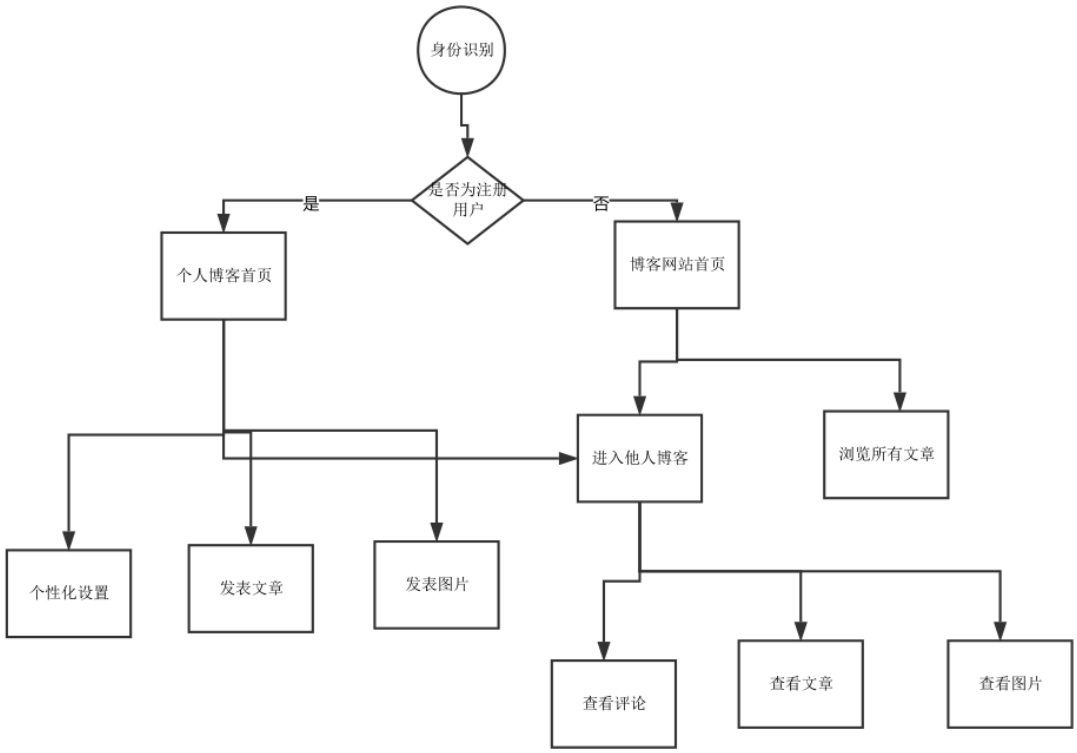

图 9 系统流程图

第四章 博客网站系统设计

4.1 博客网站系统框架

本网站采用 SSH 框架进行开发实现。

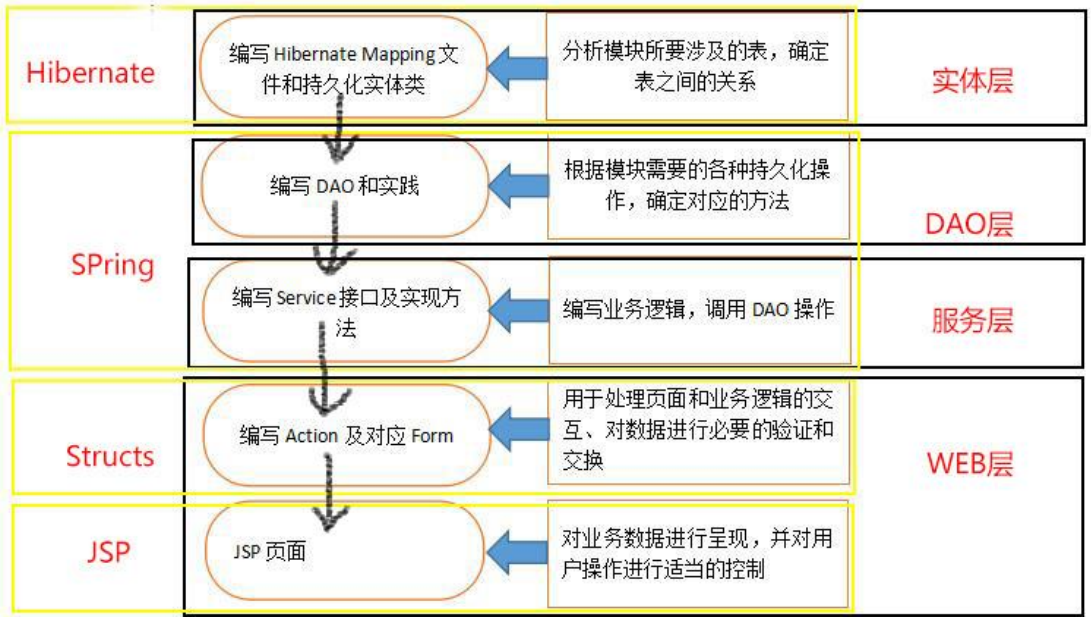

图 10 SSH 架构图

Struts 负责 Web 层：

ActionForm Bean 接收网页中表单提交的数据，然后通过 Action 进行处理，再 Forward 到对应的网页，在 Struts-config.xml 中定义了<action-mapping>，ActionServlet 会加载进来。

Spring 负责业务层管理，即 Service：

Service 为 Action 提供统一的调用接口，封装持久层的 DAO，并集成 Hibernate，Spring 可对 JavaBean 和事物进行统一管理。

Hibernate 负责持久层，完成数据库的 CRUD 操作：

Hibernate 有一组 hbm.xml 文件和 PO，是与数据库中的表相对应的，然后定义 DAO，这些是与数据库打交道的类。

在 Struts+Spring+Hibernate 系统中，对象之间的调用流程如下：

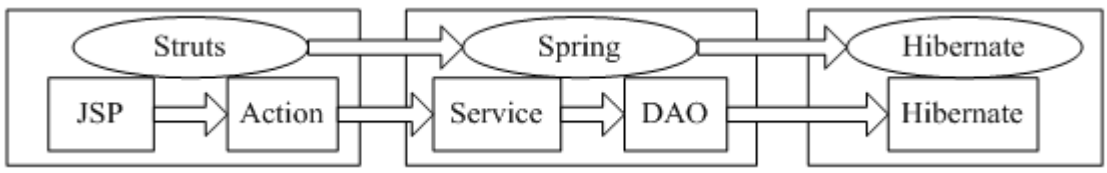

图 11 SSH 系统中对象之间的调用流程图

Struts——>Spring——>Hibernate

JSP——>Action——>Service——>DAO——>Hibernate

4.2 博客网站系统关键抽象

系统关键抽象即系统实体类图，系统实体类图描述了系统中的类及其相互之间的各种关系，它反映了系统中包含的各种对象的类型以及对象间的各种静态关系。（图 11）主要描述了系统实体层中各实体类的属性及其相互的关系。

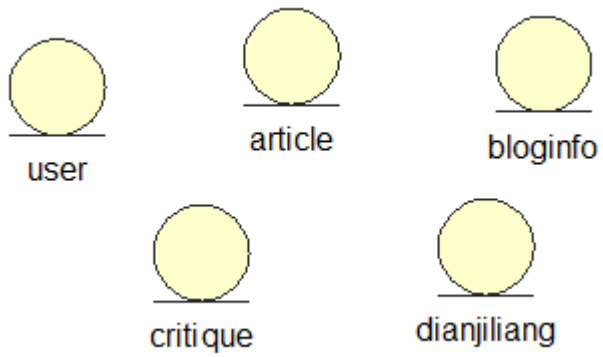

图 12 系统关键抽象

4.3 创建系统静态模型

4.3.1 创建系统类图

在系统关键抽象的基础上，添加了个实体类的属性，形成如下系统类图。

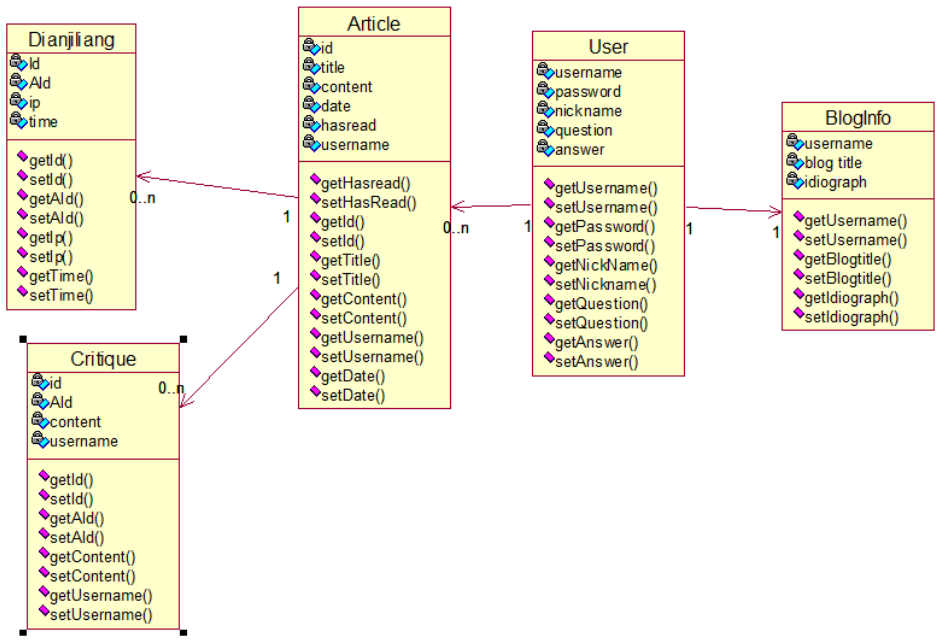

图 13 博客系统类图

4.3.2 组织系统包图

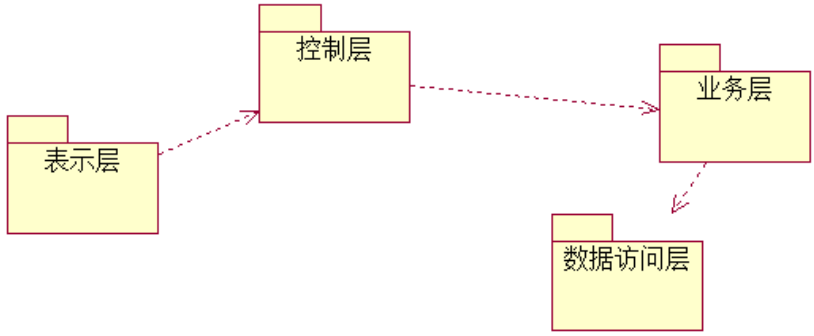

图 14 博客系统包图

4.4 数据库分析与设计

4.4.1 数据库概念设计（实体 E-R 图）

本系统一共设计规划出 5 个实体，分别是用户实体、文章实体、个性设置实体、评论实体及点击量实体。

用户实体用来保存用户所有的信息，包括用户名、密码、昵称、密保问题及密保回答，其 E-R 图如下：

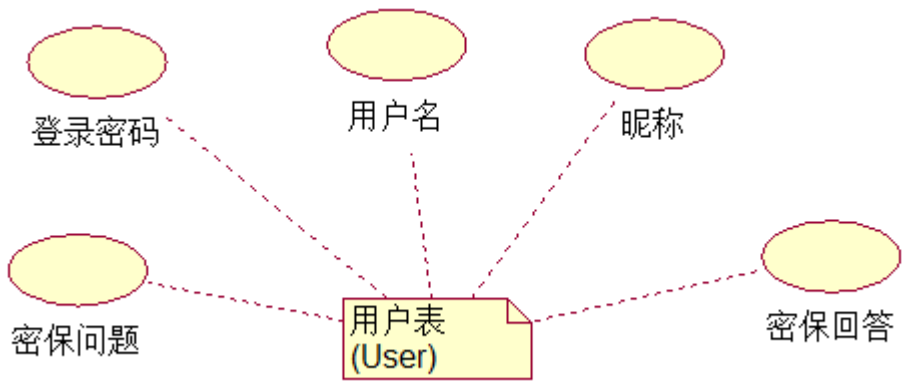

图 15 用户信息实体 E-R 图

个性设置实体用来保存博客个性化信息，包括用户名、博客标题以及个性签名信息。个性设置实体 E-R 图如下：

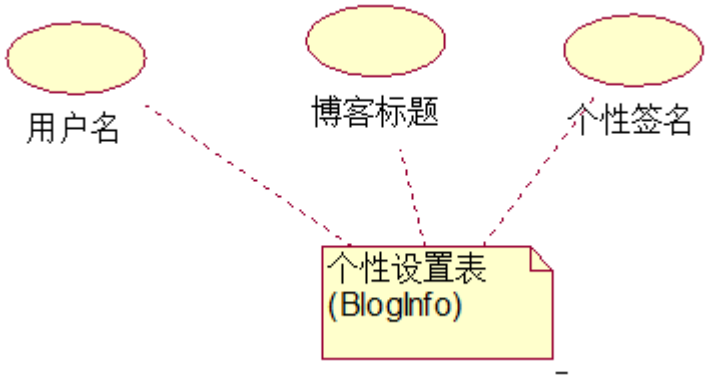

图 16 个性设置实体 E-R 图

文章实体用来保存文章的基本信息，包括文章编号、文章标题、文章内容、发表用户名、发表时间及评论数。文章实体 E-R 图如下：

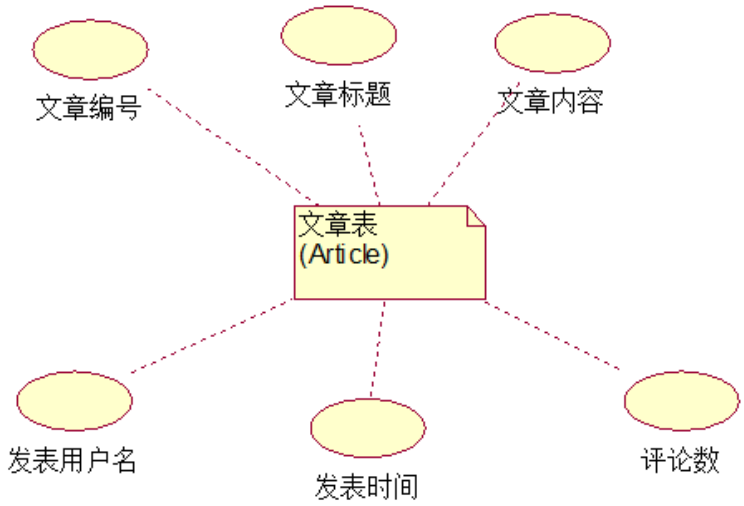

图 17 文章实体 E-R 图

评论实体用来保存评论的基本信息，包括评论编号、所属文章编号、评论内容及评论用户名。评论实体 E-R 图如下：

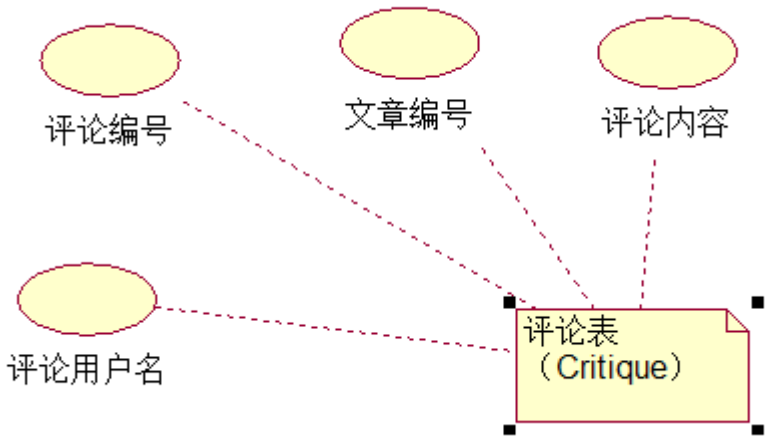

图 18 评论实体 E-R 图

点击量实体用来保存点击文章的基本信息，包括点击量编号、文章编号、点击者 ip 地址及点击时间。点击量实体 E-R 图如下：

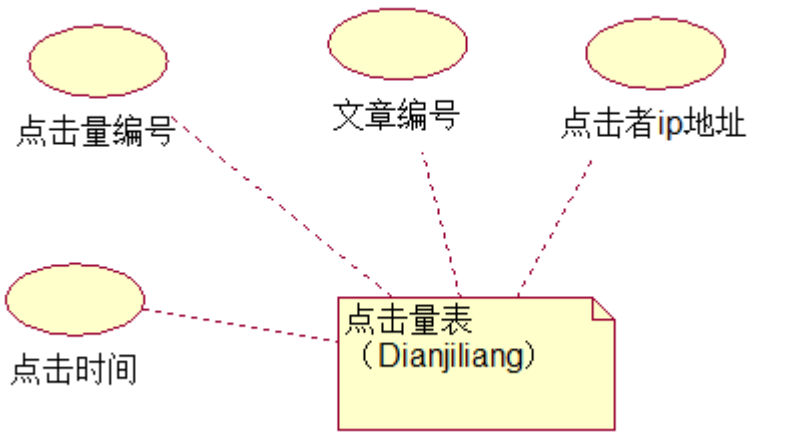

图 19 点击量实体 E-R 图

4.4.2 数据库逻辑结构设计

根据设计好的各实体 E-R 图创建数据库的逻辑结构，数据库各表的结构如下：

表 4-1 用户表

字段名	数据类型	是否主键	描述
username	文本(varchar)	是	用户名
password	文本(varchar)	否	登录密码
nickname	文本(varchar)	否	昵称
question	文本(varchar)	否	密码保护问题
answer	文本(varchar)	否	密码保护回答

表 4-2 个性设置表

字段名	数据类型	是否主键	描述
username	文本(varchar)	是	用户名
blogtitle	文本(varchar)	否	博客标题
idiograph	文本(varchar)	否	个性签名

表 4-3 文章表

字段名	数据类型	是否主键	描述
Id	整数(int)	是	文章 ID
title	文本(varchar)	否	文章标题
content	文本(varchar)	否	文章内容
username	文本(varchar)	否	发表用户
date	日期	否	发表时间
hasread	整数(int)	否	评论数

表 4-4 评论表

字段名	数据类型	是否主键	描述
Id	整数(int)	是	评论 ID
Aid	整数(int)	否	所属文章 ID
content	文本(varchar)	否	评论内容
username	文本(varchar)	否	发表用户

表 4-5 点击量表

字段名	数据类型	是否主键	描述
Id	整数(int)	是	点击 ID
Aid	整数(int)	否	所属文章 ID
ip	文本(varchar)	否	点击者 IP
time	日期	否	点击时间

4.5 系统功能模块实现

4.5.1 系统功能模块结构

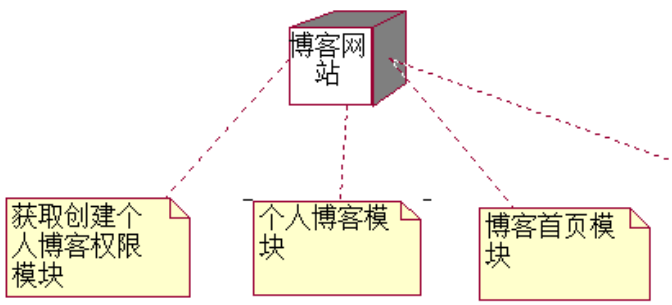

图 19 系统功能模块结构图

4.5.2 公共类设计

公共类设计包括用户信息类、文章信息类、评论信息类、博客信息类、点击量信息类、登录拦截器。

用户信息类：

```
public class User {
    private String username;
    private String password;
    private String nickname;
    private String question;
    private String answer;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
```

图 20 用户信息类 User.java

用户信息类定义了用户的用户名、密码、昵称、密保问题、密保答案属性，还为这些属性生成 Getter 和 Setter 方法。在 Hibernate 中需要对用户实体类进行配置：

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.sanqing.po.User">
        <id name="username">
            <generator class="assigned"></generator>
        </id>
        <property name="password"></property>
        <property name="nickname"></property>
        <property name="question"></property>
        <property name="answer"></property>
    </class>
</hibernate-mapping>
```

图 21 Hibernate 中配置用户实体类 User.hbm.xml

文章信息类：

```java
import java.util.Date;

public class Article {
    private int id;
    private String title;
    private String content;
    private String username;
    private Date date;
    private int hasread;

    public int getHasread() {
        return hasread;
    }
    public void setHasread(int hasread) {
        this.hasread = hasread;
    }
    public int getId() {
```

图 22 文章信息类 Article.java

文章信息类中封装了博客系统中发表文章的所有信息，包括文章 ID、文章标题、文章内容、发表用户名、发表时间和评论数量属性，并为这些属性定义了 Getter 和 Setter 方法。在 Hibernate 中需要对文章实体类进行配置：

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.sanqing.po.Article">
        <id name="id">
            <generator class="increment"></generator>
        </id>
        <property name="title"></property>
        <property name="content"></property>
        <property name="username"></property>
        <property name="date"></property>
        <property name="hasread"></property>
    </class>
</hibernate-mapping>
```

图 23 Hibernate 中配置文章实体类 Article.hbm.xml

评论信息类：

```java
public class Critique {
    private int id;
    private int AId;
    private String content;
    private String username;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getAId() {
```

图 24 评论信息类 Critique.java

评论信息类中封装了博客系统中发表评论的所有信息，包括评论 ID、评论文章、评论内容、评论用户属性，并为这些属性定义了 Getter 和 Setter 方法。在 Hibernate 中需要对文章实体类进行配置：

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.sanqing.po.Critique">
        <id name="id">
            <generator class="increment"></generator>
        </id>
        <property name="AId"></property>
        <property name="content"></property>
        <property name="username"></property>
    </class>
</hibernate-mapping>
```

图 25 Hibernate 中配置评论实体类 Critique.hbm.xml

博客信息类：

```java
package com.sanqing.po;

public class BlogInfo {
    private String username;
    private String blogtitle;
    private String idiograph;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getBlogtitle() {
        return blogtitle;
```

图 26 博客信息类 BlogInfo.java

博客信息类中封装了博客个性化信息，包括用户名、博客标题以及个性签名信息属性，并为这些属性定义了 Getter 和 Setter 方法。在 Hibernate 中需要对文章实体类进行配置：

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.sanqing.po.BlogInfo">
        <id name="username">
            <generator class="assigned"></generator>
        </id>
        <property name="blogtitle"></property>
        <property name="idiograph"></property>
    </class>
</hibernate-mapping>
```

图 27 Hibernate 中配置博客信息实体类 BlogInfo.hbm.xml

点击量信息类：

```java
package com.sanqing.po;

import java.util.Date;

public class Dianjiliang {
    private int Id;
    private int AId;
    private String ip;
    private Date time;
    public int getId() {
        return Id;
    }
    public void setId(int id) {
        Id = id;
    }
    public int getAId() {
        return AId;
    }
```

图 28 点击量信息类 Dianjiliang.java

点击量信息类中封装了点击文章的基本信息，包括点击量编号、文章编号、点击者 ip 地址及点击时间属性，并为这些属性定义了 Getter 和 Setter 方法。在 Hibernate 中需要对文章实体类进行配置：

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.sanqing.po.Dianjiliang">
        <id name="id">
            <generator class="increment"></generator>
        </id>
        <property name="AId"></property>
        <property name="ip"></property>
        <property name="time"></property>
    </class>
</hibernate-mapping>
```

图 29 Hibernate 中配置点击量信息实体类 Dianjiliang.hbm.xml

登录拦截器：

在网站中有些功能是需要用户登录才能操作的，比如博客网站中发表文章，但是因为该操作不固定位置，所以直接在该功能上加一个控制层是不合适的。所

以在此采用拦截器的概念，在操作之前先经过拦截器，通过拦截器判断用户是否登录。

```
package com.sanqing.interceptor;
import java.util.Map;
import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

public class UserInterceptor extends AbstractInterceptor {

    public String intercept(ActionInvocation invocation) throws Exception {
        ActionContext context =invocation.getInvocationContext();
        //get session
        Map session = context.getContext().getSession();
        String username = (String) session.get("username");
        //get the username in the session
        if(username == null || "".equals(username)) {
            //return log in interface
            return Action.LOGIN;
        } else {
            //go on next step
            return invocation.invoke();
        }
    }

}
```

图 30 拦截器类 UserInterceptor.java

拦截器要起到效果还需要在 Struts 配置文件 struts.xml 中进行配置：

```
<interceptors>
    <interceptor name="userInterceptor" class="
    com.sanqing.interceptor.UserInterceptor"></
    interceptor>
</interceptors>
```

图 31 在 struts.xml 进行拦截器配置

4.6 创建系统部署模型

4.6.1 构件图

对系统的主要参与者和主要业务实体类分别创建对应的构件进行映射，众多的 web 页面组成一个总的界面构件，而所有的构件又形成 web 主程序。本系统的构件图如下：

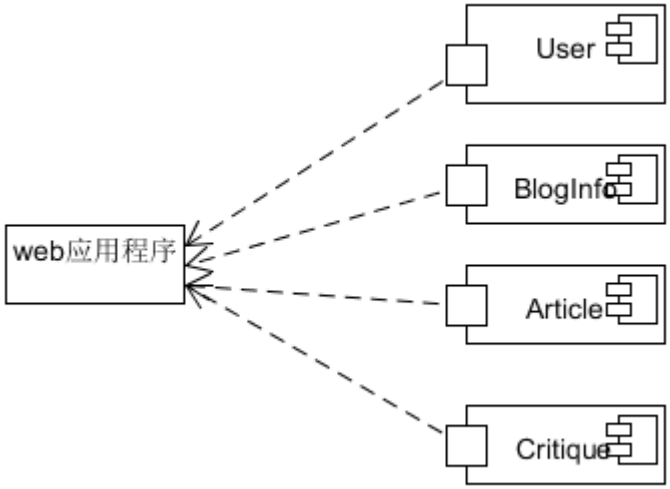

图 32 博客系统构件图

4.6.2 部署图

系统部署图描述的是系统节点上运行资源的安排。本系统包括 3 种节点：

- 数据库服务器节点：负责数据存储和处理；
- Web 系统服务器节点：发布 web 应用程序；
- Web 浏览器节点：客户端节点，用户在浏览器上进行的各种操作。

本系统的部署图如下：

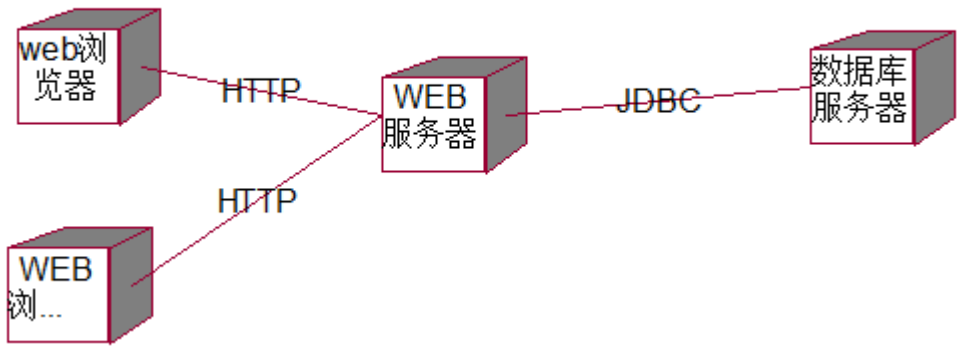

图 33 博客系统部署图