# 基于MVC模式的个人blog系统的设计与实现

**摘 要**

随着互联网技术的高速发展，生活节奏的加快，博客被越来越多的人选择为学习和交流的工具。博客的内容丰富多彩，包括对其他网站的超链接和评论、个人构思、新闻日志、照片、诗歌等。博客具有自主性、开放性、互动性和共享性的特点，是一个很好的交流的渠道。

本系统利用JSP开发技术，以MySQL作为后台数据库，使用Tomcat配置Web服务器，以及配合多种网页开发工具，实现了个人博客系统这个基于B/S模式的个人日志发表和交流的平台。本系统是一个实用的个人博客系统，集日志、相册、留言板于一身。分为管理员管理模块：负责管理所有信息；注册登录模块：注册用户、登录系统；日志管理模块：发表、修改、删除日志；评论留言管理模块：游客可以发表评论和留言；图片管理模块：用户可以创建相册，上传图片。

本文组织如下：第一章为对本系统整体的概述，第二章为对系统的详细设计，第三章为对于开发工具开发环境的选择，第四章讲的是数据库的设计，第五章则是系统完成后的测试及发布。

**关键词：**博客；JSP；MySQL；B/S模式

**课程设计开发平台与工具：** MYeclipse+ JSP+javabean+servlet +mysql

**1.课程设计任务**

本系统主要实现博客信息管理系统的开发工作，主要设计界面友好，该系统主要包含三类用户：博客访问者，管理员以及系统管理员。该系统主要包括前台管理模块和后台管理模块。博客访问者可以浏览信息，发布信息等，后台管理员通过登录后可以发布信息管理信息等。博客系统管理员负责对整个系统进行统一管理。

**2.课程设计任务（功能）说明及功能模块图**

博客信息管理系统允许用户有三种身份：博客访问者、博客管理员及系统管理员。博客访问者无需登陆就可以浏览信息、发表评论或留言；博客管理员注册后再登陆，就可以发布日志、管理个人主页；而博客管理员的基本信息都由系统管理员对其统一管理。

（1）该系统包含三种用户：分别是用户、系统管理员、博客管理员，其用例图如下：



图2.1系统用户用例图

（2）系统主要分为前台和后台管理，详细说明如下所示：

前台用户浏览模块主要用于用户浏览日志信息，用户浏览日志有三种浏览方式可以选择：一是直接点击日志标题链接进入日志显示页面浏览日志；二是点击博客主题链接进入该个人博客的首页，然后在该首页中选择所需的日志标题链接进入日志显示页面浏览日志；三是通过首页提供的搜索功能，搜索所需要的日志，然后在结果列表中选择所需的日志标题链接进入日志显示页面浏览日志。浏览日志的同时用户可以查看别人对该日志的评论，然后用户也可以对该日志发表自已的评论。

**需求分析：**

这几年，许多新名词涌入我们的视野：博客、圈子、播客、WAP等。这些都预示着我们进入了一个新的互联网阶段web 2.0，它是相对web 1.0的新的一类互联网应用的总称，是一次从核心内容到外部应用的革命。这个阶段发展迅速，互联网应用趋于多样化，其中变化最大的是由web 1.0网站创造内容的时代变为由用户创造内容的web 2.0时代。

而博客及博客文化正成为互联网的热点，并被视为继Email、BBS和ICQ之后出现的第四种网络交流方式。

博客正在改变组织沟通和社会交流方式。目前，全球参与博客的人数已经达到1000万之众，而且每5.8秒钟都有一名新的博客用户加盟进来。很多记者、律师、教授、学者、专家，甚至法官和议员都有了自己的博客网站，形成了独特的博客文化。美国总统候选人乐于开设博客站点，作为竞选的重要手段。企业界也正在把博客方式引入企业内部网和公司网站，用于非正式的知识管理和网络营销等。政府部门和教育领域都在尝试利用博客构建内部知识管理和沟通交流的新体系，可以让职员发表评论，积累知识，链接有意思的网络资源，促进彼此的交流沟通。

显然，博客的发展对应了全球范围内知识社会来临的背景。对于中国这样一个组织机构信息化仍处于初级阶段的国家，博客这种充分发挥个人主导性和积极性的非正式知识管理工具和其深度沟通交流的功能，对于我们国家的知识传播、知识扩散和有效提升组织的学习和创新能力，具有更特殊的意义。博客用户的快速增长以及组织机构对它的广泛应用，有助于推动中国向学习型社会演进。

1.2 问题的提出

博客记载了日常发生的事情和自己的兴趣爱好，把自己的思想和知识与他人分享、交流，同时又通过这个空间结识了更多志趣相投的朋友；而越来越多专业知识的Blog的出现，让我们看到了 Blog更多所蕴涵的巨大的信息价值。不同的Blog选择不同的内容，收集和整理成为让很多人关注的专业Blog——目前越来越多的人获取信息的来源是一些固定的Blog。随着博客人数的增加，Blog作为一种新的生活方式、新的工作方式、新的学习方式已经被越来越多的人所接受，并且在改变传统的网络和社会结构，网络信息不再是虚假不可验证的，交流和沟通更有明确的选择和方向性，单一的思想和群体的智慧结合变的更加有效，个人出版变成人人都可以实现的梦想——Blog正在影响和改变着我们的生活。

1.3 系统的开发目标

本次毕业设计实现了一个实用的个人博客系统，集日志、相册、留言板于一身。

本系统中管理员通过后台登录页面进入后台管理模块后，可对注册的博客用户进行维护和管理，包括对注册用户的查看和删除。

管理员登录后，还可对帐户的所有日志、图片以及评论和留言进行管理，包括查看操作、删除操作。

博客注册用户通过前台登录后，可对自己的博客空间进行管理，包括发布自己的网络日志、收藏个人图片、和相关人员进行交流和沟通等。

因此，在具体设计实现该博客网站时，主要考虑了主流博客网站的几个主要功能:(1)博客的注册、登录功能（2）管理员管理所有用户的日志、评论、留言和图片(3)日志详细内容及相关评论显示(4)博客个人日志管理维护功能(5)博客个人基本信息管理维护功能(6)博客图片上传及个人相册管理(7)网络用户写留言，博主查看留言。

## 系统的开发目标

本次毕业设计实现了一个实用的个人博客系统，集日志、相册、留言板于一身。

本系统中管理员通过后台登录页面进入后台管理模块后，可对注册的博客用户进行维护和管理，包括对注册用户的查看和删除。

管理员登录后，还可对帐户的所有日志、图片以及评论和留言进行管理，包括查看操作、删除操作。

博客注册用户通过前台登录后，可对自己的博客空间进行管理，包括发布自己的网络日志、收藏个人图片、和相关人员进行交流和沟通等。

因此，在具体设计实现该博客网站时，主要考虑了主流博客网站的几个主要功能:(1)博客的注册、登录功能（2）管理员管理所有用户的日志、评论、留言和图片(3)日志详细内容及相关评论显示(4)博客个人日志管理维护功能(5)博客个人基本信息管理维护功能(6)博客图片上传及个人相册管理(7)网络用户写留言，博主查看留言。

### 注册博友

首先必须在博客首页中登录填写用户名和密码，这样才能执行一些相关操作，不然就是普通用户只能查看一些信息，而不能发表博文及图片等。可以通过各个日志模块进行日志的添加删除修改操作，也可以上传图片和游览自己的相册，在上传过程中可以对图片的信息进行相关的描述，可以直观的让用户看到这个图片的意义。在个人管理页面中注册用户还可以修改自己的个人信息。用例图如图2.2所示。

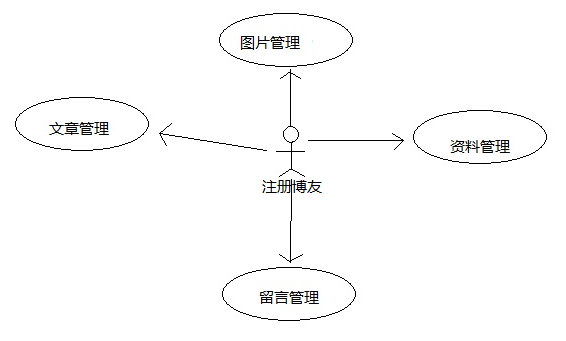
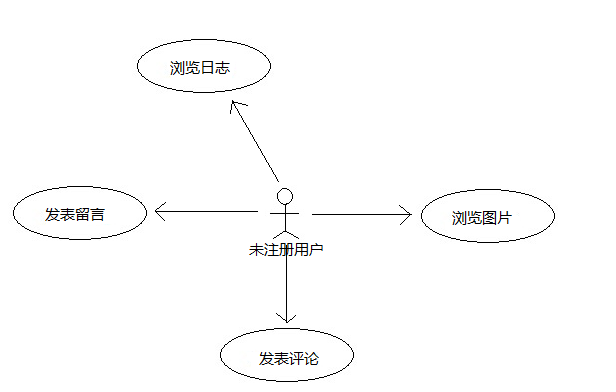


图2.2注册博友的用例图

在博客主界面中可以显示最新的博文，这样一目了然博主最新的状态。点击进入博客日志的列表后可以查看所有日志，点击进入各个日志后可以查看网友对自己的博文的一些评论，进入留言板后可以查看网友留下的一些留言，提供与其他网友交流的空间，更好交流，但不能在自己的博客主页面中提交评论和留言。通过这些功能这样就可以在本系统中为博友提供更多的个人色彩。

### 普通未注册用户

在博客注册页面注册成为博客的注册用户，才可以拥有自己独立的空间。而未注册用户进入某一博客页面查看相关日志信息时，可以随时对自己所感兴趣的日志发表评论，同时也可以查看到其他人针对该博文的评论，普通未注册用户也可以在阅读博文时最后留下自己的一些评论。他也可以留下更多的信息，例如可以在留言板给博主留言，但这时普通未注册用户的留言和评论显示的将是名为游客的评论而不是博客注册的昵称。在博客主界面中可以显示最新的博文，这样一目了然博主最新的状态。点击进入博客日志的列表后可以查看所有日志，点击进入各个日志后可以查看网友对自己的博文的一些评论，进入留言板后可以查看网友留下的一些留言，提供与其他网友交流的空间，更好交流，但不能在自己的博客主页面中提交评论和留言。通过这些功能这样就可以在本系统中为博友提供：



## 未注册用户博客系统中的用例图

## 功能模块介绍

### 博客注册登录管理模块

博客注册登录管理模块用于建立博客网站固定的客户群体，通过记录对应的博客档案，实现对博客信息的维护及管理，同时也便于通过博客档案将网站最新动态及相关信息方便地传达给每一位浏览的用户。

该功能模块实现了以下几个子功能。

（1）新博客在线注册。

（2）博客主登录。

（3）跳转到博客主页。

只有进行登录并通过身份验证的用户，才可以在个人博客页面发表日志，并借助个人设置实现对个人博客相关信息的管理维护。对于没有经过身份验证的网络用户不允许在博客页面中发表日志，更不允许对博客页面信息进行管理维护。

该模块实现新博客的注册及登录验证功能。其中，注册新博客时会对用户输入的注册信息进行有效性验证，包括基本数据格式的有效性以及逻辑有效性。注册成功的博客主登录时，会随时根据博客输入的登录信息进行提示，如用户名错误或者密码错误。

### 博客日志模块

日志模块为网络注册用户提供便捷的管理功能，可以添加、删除、修改日志，以及日志阅读浏览等功能，同时日志的评论信息也能够及时反馈给网络用户。

该功能模块实现了以下几个子功能。

（1）添加博客日志

发表博客日志主要是提供用户发表和张贴个人的日志或是对某些事务的独到见解等的功能，通过发表的日志，其他用户可以发表相关的评论，以便能够同其他来访者进行交流，是一个展现自己的网络交流平台。

该功能通过日志的Servlet来控制完成并写入到数据库中。在保存过程中，系统会在数据库中为该日志自动生成一个唯一的日志ID编号，这个编号是自

## 功能模块介绍

### 博客注册登录管理模块

博客注册登录管理模块用于建立博客网站固定的客户群体，通过记录对应的博客档案，实现对博客信息的维护及管理，同时也便于通过博客档案将网站最新动态及相关信息方便地传达给每一位浏览的用户。

该功能模块实现了以下几个子功能。

（1）新博客在线注册。

（2）博客主登录。

（3）跳转到博客主页。

只有进行登录并通过身份验证的用户，才可以在个人博客页面发表日志，并借助个人设置实现对个人博客相关信息的管理维护。对于没有经过身份验证的网络用户不允许在博客页面中发表日志，更不允许对博客页面信息进行管理维护。

该模块实现新博客的注册及登录验证功能。其中，注册新博客时会对用户输入的注册信息进行有效性验证，包括基本数据格式的有效性以及逻辑有效性。注册成功的博客主登录时，会随时根据博客输入的登录信息进行提示，如用户名错误或者密码错误。

### 博客日志模块

日志模块为网络注册用户提供便捷的管理功能，可以添加、删除、修改日志，以及日志阅读浏览等功能，同时日志的评论信息也能够及时反馈给网络用户。

该功能模块实现了以下几个子功能。

（1）添加博客日志

发表博客日志主要是提供用户发表和张贴个人的日志或是对某些事务的独到见解等的功能，通过发表的日志，其他用户可以发表相关的评论，以便能够同其他来访者进行交流，是一个展现自己的网络交流平台。

该功能通过日志的Servlet来控制完成并写入到数据库中。在保存过程中，系统会在数据库中为该日志自动生成一个唯一的日志ID编号，这个编号是自动递增的生成，用以唯一的标识一篇日志。

（2）单独日志的显示，评论查看和发表评论

发表日志的评论是在查看单篇日志的前提下操作的，当访客查看单独的一篇日志时，即可通过页面下方的评论输入框来进行评论的发布。

该模块能够在网页中随时提供在线的最新日志信息。网络用户可以随时获得最新日志以及最新的博客评论。在用户选择了某个感兴趣的日志后，可以方便地跳转到对应博客页面进行日志的阅读，并和博客主进行交流互动。

### 博客图片模块

当网络用户进入某个人博客相关相册页面后，在该页面中将提供博客图片列表的显示，也提供对单个图片的浏览及评论的浏览。该功能模块实现了如下几个子功能。

1. 添加相册和图片。

进入相册模块后，可以点击创建相册来新建一个相册，建立相册后，可以进入相册模块点击上传按钮进行相关的上传操作。上传的文件将被保存在IMAGE的文件夹中。图片将以存放的相对地址存入数据库中，以便保存和读取。

1. 浏览图片。

可以进入相应的相册进行相册图片的查看，这时所有该相册的图片将会显示在浏览器中，通过单击相应的图片可以进入单个图片的查看页面。

1. 浏览单个图片及发表评论。

### 发表图片的评论是在查看单个图片的前提下操作的，当访客查看单独图片时博客留言管理模块

博客留言管理模块用于实现用户对博客个人留言的管理和显示。

该功能模块实现了如下几个子功能：

（1）查看留言。

（2）各种用户发表留言。

借助该模块，允许用户对博客的留言进行查看及其发表。

### 博客个人维护管理模块

博客个人维护管理模块用于实现用户对博客个人主页及相关信息的动态管理。

该功能模块实现了如下几个子功能：

（1）个人注册信息的修改，密码的修改等。

（2）个人基本信息维护管理。

借助该模块，允许用户对博客的个人信息进行维护及其管理。

### 博客后台管理模块

在博客后台管理模块中博客系统的管理员可以登陆后台进行整个博客系统的管理，进入某一管理页面可以查看相关方面的信息，可以随时对相关信息进行人为审查，如评论，日志中是否有敏感词，博客主上传的图片是否含有暴力色情等等，同时也可以查看到其他人的评论和留言中是否含有人身攻击等。

该功能模块实现了如下几个子功能：

（1）对各个分类如日志、图片、评论的显示。

（2）对各个相关信息单独的显示。

（3）对不符合规定的相关文件进行删除。

借助该模块，允许管理员对博客的各个信息进行维护及其管理。

2.3 系统分析

在整个blog进行开发之前，要确定出整个项目的整体架构，包括系统的选型、运行环境的确定及系统结构设计。下面对这进行详细介绍。

在进行软件系统开发的最初环节，一般都需要进行系统的选型，即根据系统功能的实际需求，选择合适的开发工具及软件架构。

blog对系统的可靠性、稳定性有比较高的要求。本系统设计时，比较主流的B/S设计有基于JSP、ASP、PHP、CGI及J2EE等模式。相比较而言PHP的功能相对简单，不适合做大程序；而CGI效率相对较低，所以也不考虑。所以选择Java技术作为blog 的开发语言。

为了增加系统的吞吐量，提高并发处理客户请求数量，系统采用了高性能PC作为主机。在数据库处理方面，不需要在数据层借助存储过程及数据库服务器端函数封装过多的业务逻辑，因此数据库系统采用相对精巧的MySQL。

该在线博客系统服务器端如果需要布置到其他主机上，则该主机必备条件如下：

1. 服务器端操作系统：独立于操作系统的跨平台系统，客户端MicroSoft Windows 2000及以上；

2. 数据库：MySQL 5.0.27版本；

3. Web服务器：Tomcat 5.5及以上版本，配合MVC设计模式；

4. 客户端运行环境：能运行IE 5以上或Netscape 5以上浏览器的操作系统；

5. 客户端运行工具：目前的系统采用浏览器作为客户端，应该选择使用IE 5以上版本浏览器。

本网站以windows xp为Web平台，JSP+Servlet+JavaBean为网站实现技术，建立基于MySQL数据库系统的核心动态网页，实现博客网站前台及博客个人维护管理等功能模块。

1、系统处理的准确性和及时性：系统处理的准确性和及时性是系统的必要性能。在系统设计和开发过程中，要充分考虑系统当前和将来可能承受的工作量，使系统的处理能力和响应时间能够满足信息处理的需求。

2、系统的开放性和系统的可扩充性：系统在开发过程中，应该充分考虑以后的可扩充性。例如数据表中用户选择字段方式的改变，用户查询的需求也会不断的更新和完善。所有这些，都要求系统提供足够的手段进行功能的调整和扩充。而要实现这一点，应通过系统的开放性来完成，既系统应是一个开放系统，只要符合一定的规范，可以简单的加入和减少系统的模块，配置系统的硬件。通过软件的修补、替换完成系统的升级和更新换代。

3、系统的易用性和易维护性：要实现这一点，就要求系统应该尽量使用用户熟悉的术语和中文信息的界面；针对用户可能出现的使用问题，要提供足够的在线帮助，缩短用户对系统熟悉的过程。

4、系统的数据要求：1、数据录入和处理的准确性和实时性。2、数据的一致性与完整性。3、数据的共享与独立性。

本系统采用MVC模式，主要有如下几个分层。

1. 表现层:由JSP 页面组成。

2. 中间层:使用Servlet和DAO。

3. 模型层:由各个功能模块相应的JAVABEAN文件对应实现。

2.3.1 系统架构说明

本系统严格按MVC 模式设计，分为控制层，表示层，模型层三层，MVC 层的控制器不进行持久化操作，只参与业务逻辑的实现，持久化的操作通过DAO来实现，表现层采用传统JSP 技术，模型层则用JAVABEAN来实现。

本系统采用的是典型的J2EE 三层结构，分为表现层、中间层(业务逻辑层)和数据服务层。三层体系将业务规则、数据访问等工作放在中间层处理。客户端不直接与数据库交互，而是通过组件与中间层建立连接，再由中间层与数据库交互。Jsp广泛的应用和稳定的表现，为其作为表现层技术打下了坚实的基础。中间层采用的是流行的Servlet和DAO技术来实现 。该系统中间层的介绍如下：这层实际上就是MVC 模式里面的C，负责模型层和表现层的交互。调用DAO，并将业务数据返回给表现层作组织表现。以DAO为基础，通过对DAO 组件的正面模式包装，完成系统所要求的业务逻辑。DAO就负责与持久化对象交互。该层封装了数据的增、删、查、改的操作。



图2.2前台管理结构图

用户进入后台管理平台，首先必需进行权限验证，防止未授权的用户非法登录。若未通过验证，则显示出错提示，并跳转到登录界面。若通过验证，则进入个人博客后台管理平台。用户可以浏览已发表的日志，并对其进行相应的操作如修改日志和删除日志；可以查看日志的评论信息并对其进行回复或者删除操作；可以发表新的日志；可以新建日志分类；可以浏览个人博客的留言信息并对其进行相应的操作如回复留言。



图2.3后台管理结构图

（3）前台模块功能结构如图：



图2.4前台功能模块图

（4）后台模块功能结构图：



图2.5后台结构图

**3. 数据库设计**

**4.各模块的功能说明**

博客信息管理系统分为面向网络用户的前台及面向个人博客维护管理的后台，依据博客网站中所要实现的基本功能的设定，将所有功能分解为如下四个主要的功能模块来加以实现。用户以合法的用户名及密码登录，系统即可按该登录用户的权限分配操作模块。

（1）博客注册登录管理模块

博客注册登录管理模块用于建立博客网站固定的客户群体，通过记录对应的博客档案，实现对博客信息的后台维护及管理，同时也便于通过博客档案库将网站最新动态及相关企业的信息方便地传达给每一位潜在的客户。该模块主要包含以下三个子模块

a）新博客在线注册。

b）博客登录管理。

c）跳转到博客主页。

只有进行登录并通过身份验证的用户，才可以在个人博客页面发表日志，并借助个人设置实现对个人博客相关信息的管理维护。对于没有经过身份验证的网络用户不允许在博客页面中发表日志，更不允许对博客页面信息进行管理维护。

（2）博客及文章检索查询模块

博客及文章检索查询模块为网络用户提供便捷的搜索，以及日志阅读浏览等功能，同时对日志的评论信息、博客推荐也能够及时反馈给网络用户。

该功能模块实现了以下几个子功能。

a）热门博客页面推荐。

b）最新博客日志推荐。

c）日志信息关键字搜索。

该模块能够在网页中随时提供在线的最新日志信息。该信息需要定期更新，网络用户可以随时获得最新日志以及最热门的博客推荐。在客户选择了某个博客或者某个感兴趣的日志后，可以方便地跳转到对应博客页面进行日志的阅读，并和博客进行交流互动。

（3）博客页面显示模块

当网络用户进入某个人博客主页后，在该页面中将提供博客日志列表的显示，同时为了方便用户浏览，在该模块中提供根据分类名进行日志列表的显示，也提供日志评论的浏览，此外还允许用户在博客页面中发表评论及留言。

该功能模块实现了如下几个子功能。

a）用户可以分页查看对应的日志内容及评论信息。

b）用户可以针对日志内容发表评论。

c）用户可以针对博客进行留言。

d）用户可以分类查看日志内容。

在该模块中还提供了博客页面统计信息，日志、评论及留言信息分页显示等方便用户的显示效果。

（4）博客个人管理维护模块

博客个人维护管理模块用于实现用户对博客个人主页及相关信息的动态管理。

该功能模块实现了如下几个子功能：

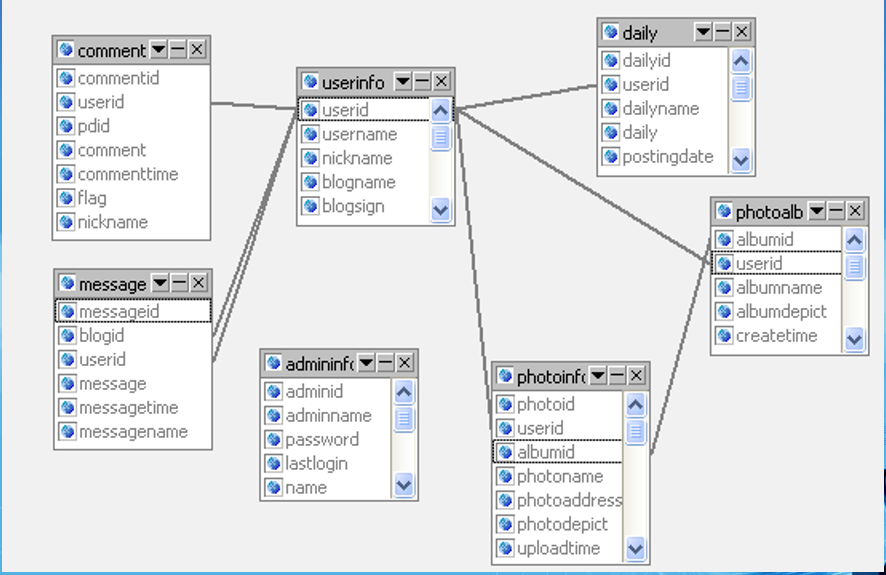
a）日志及日志分类管理。

b）评论及留言管理。

c）个人基本信息维护管理。

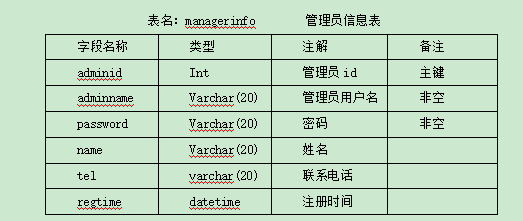
借助该模块，用户可以随时对个人博客主页中的内容进行增加或修改，包括日志分类信息的更新、评论及留言管理等功能，也允许用户对博客的个人信息进行维护及其管理。

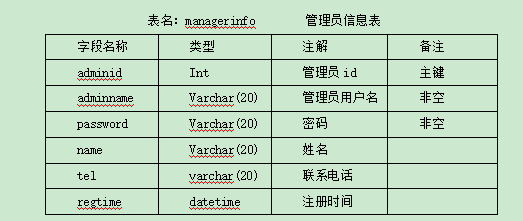
关系数据库：



数据库表的设计：

















## 系统的发布和运行效果

### 系统的发布

如果本地工作站作为服务器，则在安装了相关组件和TOMCAT以后，按照前面所介绍的进行正确的配置，申请到相应的域名后，让本地服务器连接到网络，就可以通过网络访问该系统。

如果是利用网络上的主页空间进行发布，则需要申请到可以支持JSP和MySQL数据库的Web空间，把系统所有文件上传至网络服务器上，通过空间提供商所提供的域名就可以访问到本系统。

### 系统测试

系统测试是软件开发中不可缺少的一个环节，它是衡量我们开发系统的一个标尺。它的目标是发现系统中的错误，然后通过调试对其进行修正，从而开发出高质量的完全符合用户需要的软件。

对本系统的测试主要包括以下几个方面：

（1）链接测试

一个好的系统应该不包含中断链接，花费时间检查链接可以避免错误信息的发生率。链接测试可以分为三个方面：首先，测试所有链接是否按指示的那样确实链接到了该链接的页面；其次，测试所链接的页面是否存在；最后，保证Web应用系统尚没有孤立的页面（没有链接指向该页面）。

经过系统链接测试，已经把系统中存在的所有错误链接都修改正确了，用户可以放心使用，不会造成功能上的缺失，使用上中断的感觉。

（2）检查布局的一致性

对于一个好的系统，布局的一致性也是十分重要的。我们在设计系统时，常常会因为追求单个页面的美观而忽略了系统的整体效果。这就需要对系统进行整体浏览，检查布局和图像的一致性。

通过对系统的整体浏览调试，已经可以保证系统整体布局的一致性，相信能够为用户提供简洁美观的页面，使用、观看起来很舒服。

系统维护

本系统在运行过程中可能会伴随出现各种各样的软硬件问题，这就需要进行系统的软件与硬件维护。

1.系统的硬件维护

硬件系统的维护有专职的硬件人员承担，维护分为以下两种：

定期的预防性维护，如周末或月末对服务器进行的例行检查与保养。

突发性的故障维修，由专职人员过厂商进行，这种维护也有时间要求，以免影响系统的正常工作。

由于本系统是暂时是基于单机建立的，所以只需维持本机的稳定即可，还不需进行系统硬件维护，当应用开发完毕上线后，须进行相关维护。

2.系统的软件维护

系统的软件维护主要包括程序的维护，数据文件的维护，代码的维护。本系统采用Mays为数据库，客户端应用程序采用JSP编写，系统的软件维护自然跟其有着密切的关系。

系统的运行需要有专门人员负责进行系统的正确性维护，防止因用户或管理人员的非法操作带来的系统错误；随着学校环境和业务量的增减，系统可能会不适应当时情况，这时就需要进行系统的完善性维护。除此之外，还要进行系统的预防性维护。

由于在编写本系统时，特别注意了代码的规范化，在命名规则，程序书写规范上做了很大的努力，这对将来的代码维护带来了方便。

总之，在系统正式投入运行之后，为了让系统长期高效稳定地运行工作，必须加强对系统运行的日常管理和维护

系统维护

本系统在运行过程中可能会伴随出现各种各样的软硬件问题，这就需要进行系统的软件与硬件维护。

1.系统的硬件维护

硬件系统的维护有专职的硬件人员承担，维护分为以下两种：

定期的预防性维护，如周末或月末对服务器进行的例行检查与保养。

突发性的故障维修，由专职人员过厂商进行，这种维护也有时间要求，以免影响系统的正常工作。

由于本系统是暂时是基于单机建立的，所以只需维持本机的稳定即可，还不需进行系统硬件维护，当应用开发完毕上线后，须进行相关维护。

2.系统的软件维护

系统的软件维护主要包括程序的维护，数据文件的维护，代码的维护。本系统采用Mays为数据库，客户端应用程序采用JSP编写，系统的软件维护自然跟其有着密切的关系。

系统的运行需要有专门人员负责进行系统的正确性维护，防止因用户或管理人员的非法操作带来的系统错误；随着学校环境和业务量的增减，系统可能会不适应当时情况，这时就需要进行系统的完善性维护。除此之外，还要进行系统的预防性维护。

由于在编写本系统时，特别注意了代码的规范化，在命名规则，程序书写规范上做了很大的努力，这对将来的代码维护带来了方便。

总之，在系统正式投入运行之后，为了让系统长期高效稳定地运行工作，必须加强对系统运行的日常管理和维护。

总结

经过两周紧张而又充实的实习学到了很多关于Jsp、MVC、WEB设计的知识，了解了很多书本里学不到的东西，

对以前所学得的知识进行了综合性的应用，这对今后的工作是十分有帮助的。从开始接到论文题目到系统的实现，再到论文文章的完成，一步对我来说都是新的尝试与挑战，这也是我在大学期间完成的最大的项目。在这段时间里，我学到了很多知识也有很多感受，从对博客一无所知，对jsp，servlet等相关技术很不了解的状态，我开始了漫长的学习，查看相关的资料和书籍，让自己头脑中模糊的概念逐渐清晰，使自己非常稚嫩的作品一步步完善起来，每一次改进都是我学习的收获，每一次试验的成功都会让我兴奋好一段时间。从中我也充分认识到了博客这一方式给我们生活带来的乐趣，在属于自己的网络空间上，尽情宣泄自己的情感，表达自己的感受，并且把自己的想法与他人分享。这次项目设计从建立需求分析、画用例图到建立数据库、详细代码设计都是自己一步步按照真正实际项目开发中的步骤进行的，对自己的锻炼是非常大的，是真正的自己学习的过程和研究的过程。希望这次的经历能让我在以后学习中激励我继续进步。由于时间仓促，一些技术和设备所限，该系统存在一些不足之处，有待进一步改进和完善。在论文结束之际，首先要感谢我的指导老师XXXX，在我的论文设计以及实现过程中均给了我大量的帮助和指导，她严谨的敬业精神和治学态度给我留下了深刻的印象。从她身上，我学到了许多能使我受益终身的东西，

再次对X老师表示衷心的感谢。

感谢计算机学院领导以及老师给我们创造的良好的学习环境和诸多帮助。

附录：

计算机程序

//评论的DAO

package com.blog.system.Dao;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import com.blog.util.db.\*;

import com.blog.system.Dto.CommentBean;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.UserBean;

public class CommentDao {

private Connection con;

CommentBean cb = new CommentBean();

public void setComment(CommentBean cb) {

this.cb = cb;

}

public List<CommentBean> getComment(int page, int size) throws Exception {

List<CommentBean> ret = new ArrayList<CommentBean>();

int begin = (page - 1) \* size;

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select \* from comment order by commentid limit "

+ begin + "," + size);

while (result.next()) {

int commentid = result.getInt("commentid");

String comment = result.getString("comment");

java.sql.Date commenttime = result.getDate("commenttime");

int flag = result.getInt("flag");

CommentBean cb = new CommentBean();

cb.setComment(comment);

cb.setCommenttime(commenttime);

cb.setCommentid(commentid);

cb.setFlag(flag);

ret.add(cb);

}

result.close();

stm.close();

con.close();

return ret;

}

public Long getCommentCount() {

long l = 0;

con = GetDbCon.getDbConn();

try {

Statement stm = con.createStatement();

ResultSet result = stm.executeQuery("select count(\*) from comment");

if (result.next()) {

l = result.getLong(1);

}

} catch (SQLException e) {

e.printStackTrace();

}

return l;

}

public void delComment(String commentid) throws SQLException {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

stm.execute("DELETE FROM comment where commentid ='" + commentid + "'");

}

public List<CommentBean> getTypeComment(String id, int type, int page,

int size) throws Exception {

List<CommentBean> ret = new ArrayList<CommentBean>();

int begin = (page - 1) \* size;

int id1 = Integer.parseInt(id);

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select \* from userinfo right join comment on userinfo.userid=comment.userid where pdid="

+ id1

+ " and flag="

+ type

+ " order by commentid desc limit "

+ begin

+ ","

+ size);

while (result.next()) {

int userid = result.getInt("userid");

int commentid = result.getInt("commentid");

String nickname = result.getString("nickname");

String comment = result.getString("comment");

java.sql.Date commenttime = result.getDate("commenttime");

CommentBean cb = new CommentBean();

cb.setUserid(userid);

cb.setComment(comment);

cb.setCommenttime(commenttime);

cb.setCommentid(commentid);

cb.setCommentname(nickname);

ret.add(cb);

}

result.close();

stm.close();

con.close();

return ret;

}

public Long getTypeCommentCount(String id, int type) {

long l = 0;

con = GetDbCon.getDbConn();

int id1 = Integer.parseInt(id);

try {

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select count(\*) from comment where pdid="

+ id1 + " and flag=" + type);

if (result.next()) {

l = result.getLong(1);

}

} catch (SQLException e) {

e.printStackTrace();

}

return l;

}

public void addComment(CommentBean cb, int type) {

try {

con = GetDbCon.getDbConn();

PreparedStatement stm = con

.prepareStatement("insert into comment(userid,pdid,comment,commenttime,flag) values(?,?,?,NOW(),?)");

stm.setInt(1, cb.getUserid());

stm.setInt(2, cb.getPdid());

stm.setString(3, cb.getComment());

stm.setInt(4, type);

stm.execute();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public CommentBean getSingleComment(String commentid) {

CommentBean cb = new CommentBean();

try {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

int commentid1 = Integer.parseInt(commentid);

ResultSet result = stm

.executeQuery("select \* from comment where commentid = "

+ commentid1);

while (result.next()) {

String comment = result.getString("comment");

int flag = result.getInt("flag");

java.sql.Date commenttime = result.getDate("commenttime");

cb.setCommentid(commentid1);

cb.setComment(comment);

cb.setCommenttime(commenttime);

cb.setFlag(flag);

}

result.close();

stm.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return cb;

}

}

//前台日志的Servlet

package com.blog.system.action;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import javax.servlet.\*;

import javax.servlet.http.\*;

import com.blog.system.Dao.CommentDao;

import com.blog.system.Dao.DailyDao;

import com.blog.system.Dao.UserDao;

import com.blog.system.Dto.CommentBean;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.UserBean;

public class DailyServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

String param = request.getParameter("param");

if (param == null) {

param = "view";

}

if (param == "add" || param.equals("add")) {

String dailyname = request.getParameter("dailyname");

String daily = request.getParameter("content");

DailyBean db = new DailyBean();

int userid1 = 1;

db.setUserid(userid1);

db.setDailyname(dailyname);

db.setDaily(daily);

DailyDao dd = new DailyDao();

try {

dd.addDaily(db);

} catch (Exception e) {

e.printStackTrace();

response.sendRedirect("/index.html");

}

response.sendRedirect(request.getContextPath()

+ "/use/dailydo?userid=" + userid1);

} else if (param == "view" || param.equals("view")) {

getDaily(request, response);

} else if (param == "del" || param.equals("del")) {

try {

String dailyid = request.getParameter("dailyid");

DailyDao dd = new DailyDao();

dd.delDaily(dailyid);

} catch (Exception e) {

e.printStackTrace();

}

getDaily(request, response);

} else if (param == "singledaily" || param.equals("singledaily")) {

String dailyid = request.getParameter("dailyid");

DailyDao dd = new DailyDao();

DailyBean db = dd.getSingleDaily(dailyid);

request.setAttribute("singledaily", db);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/system/singledaily.jsp");

requestDispatcher.forward(request, response);

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

public void getDaily(HttpServletRequest request,

HttpServletResponse response) throws IOException, ServletException {

int page;

if (request.getParameter("page") != null) {

page = Integer.parseInt(request.getParameter("page"));

} else {

page = 1;

}

int size = 10;

List<DailyBean> ret = new ArrayList<DailyBean>();

DailyDao dd = new DailyDao();

try {

ret = dd.getSystemDaily(page, size);

long count = dd.getDailyCount();

request.setAttribute("daily", ret);

request.setAttribute("count", count);

request.setAttribute("page", page);

request.setAttribute("size", size);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/system/systemdaily.jsp");

requestDispatcher.forward(request, response);

} catch (Exception e) {

e.printStackTrace();

}

}

}

//评论的javabean

package com.blog.system.Dto;

public class CommentBean {

private int commentid;

private int userid;

private int pdid;

private String commentname;

private String comment;

private java.sql.Date commenttime;

private int flag;

public int getCommentid() {

return commentid;

}

public void setCommentid(int commentid) {

this.commentid = commentid;

}

public int getUserid() {

return userid;

}

public void setUserid(int userid) {

this.userid = userid;

}

public int getPdid() {

return pdid;

}

public void setPdid(int pdid) {

this.pdid = pdid;

}

public String getComment() {

return comment;

}

public void setComment(String comment) {

this.comment = comment;

}

public java.sql.Date getCommenttime() {

return commenttime;

}

public void setCommenttime(java.sql.Date commenttime) {

this.commenttime = commenttime;

}

public int getFlag() {

return flag;

}

public void setFlag(int flag) {

this.flag = flag;

}

public String getCommentname() {

return commentname;

}

public void setCommentname(String commentname) {

this.commentname = commentname;

}

}

//前台图片的Servlet

package com.blog.use.action;

import java.io.\*;

import java.util.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import com.blog.system.Dao.\*;

import com.blog.system.Dto.\*;

public class PhotoServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

String param = request.getParameter("param");

if (param == null) {

param = "view";

}

if (param == "view" || param.equals("view")) {

getPhoto(request, response);

} else if (param == "del" || param.equals("del")) {

try {

String photoid = request.getParameter("photoid");

PhotoDao pd = new PhotoDao();

pd.delPhoto(photoid);

} catch (Exception e) {

e.printStackTrace();

}

getPhoto(request, response);

} else if (param == "singlephoto" || param.equals("singlephoto")) {

String photoid = request.getParameter("photoid");

PhotoDao pd = new PhotoDao();

PhotoBean pb = pd.getSinglePhoto(photoid);

request.setAttribute("singlephoto", pb);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/system/singlephoto.jsp");

requestDispatcher.forward(request, response);

}else if (param == "preupload" || param.equals("preupload")) {

String userid = request.getParameter("userid");

UserDao ud = new UserDao();

UserBean ub = ud.getSingleUser(userid);

request.setAttribute("userinfo", ub);

request.setAttribute("userid", userid);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/upload.jsp");

requestDispatcher.forward(request, response);

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

public void getPhoto(HttpServletRequest request,

HttpServletResponse response) throws IOException, ServletException {

request.setCharacterEncoding("utf-8");

int page;

if (request.getParameter("page") != null) {

page = Integer.parseInt(request.getParameter("page"));

} else {

page = 1;

}

int size = 10;

String userid = request.getParameter("userid");

String albumid = request.getParameter("albumid");

int albumid1 = Integer.parseInt(albumid);

try {

List<PhotoBean> ret = new ArrayList<PhotoBean>();

PhotoDao pd = new PhotoDao();

ret = pd.getPhoto(page, size, userid, albumid);

long count = pd.getPhotoCount();

UserDao ud = new UserDao();

UserBean ub = ud.getSingleUser(userid);

PhotoalbumDao pad = new PhotoalbumDao();

String albumname = pad.getAlbumname(albumid);

// System.out.println(albumname);

request.setAttribute("userinfo", ub);

request.setAttribute("userid", userid);

request.setAttribute("count", count);

request.setAttribute("page", page);

request.setAttribute("size", size);

request.setAttribute("photo", ret);

request.setAttribute("albumname", albumname);

request.setAttribute("albumid", albumid1);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/photo\_list.jsp");

requestDispatcher.forward(request, response);

} catch (Exception e) {

e.printStackTrace();

}

}

}

//图片上传的Servlet

package com.blog.use.action;

import java.io.IOException;

import java.net.URLDecoder;

import java.util.Enumeration;

import javax.servlet.ServletConfig;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.blog.system.Dao.DailyDao;

import com.blog.system.Dao.PhotoDao;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.PhotoBean;

import com.jspsmart.upload.File;

import com.jspsmart.upload.SmartUpload;

import com.jspsmart.upload.SmartUploadException;

public class UploadServlet extends HttpServlet {

private ServletConfig config;

final public void init(ServletConfig config) throws ServletException {

this.config = config;

}

final public ServletConfig getServletConfig() {

return config;

}

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doPost(request, response);

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("gb2312");

try {

SmartUpload su = new SmartUpload();

su.initialize(config, request, response);

su.setMaxFileSize(5 \* 1024 \* 1024);

su.save("/image/");

su.upload();

File file = su.getFiles().getFile(0);

String picName = (file.getFileName());

file.saveAs("/image/" + picName, su.SAVE\_VIRTUAL);

String photoname = su.getRequest().getParameter("photoname");

String photodepict = su.getRequest().getParameter("photodepict");

String serverPath = request.getSession().getServletContext()

.getRealPath("");

String photoaddress = request.getContextPath()+"/image/" + picName;

PhotoBean pb = new PhotoBean();

//int viewuserid = (Integer)request.getSession().getAttribute("viewuserid");

int albumid = (Integer)request.getSession().getAttribute("albumid");

//System.out.println(albumid);

int userid = 1;

pb.setUserid(userid);

pb.setPhotoname(photoname);

pb.setPhotodepict(photodepict);

pb.setPhotoaddress(photoaddress);

pb.setAlbumid(albumid);

PhotoDao pd = new PhotoDao();

pd.addPhoto(pb);

} catch (SmartUploadException e) {

e.printStackTrace();

}

}

}

前台单个日志及评论的Servlet

package com.blog.use.action;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.blog.system.Dao.CommentDao;

import com.blog.system.Dao.DailyDao;

import com.blog.system.Dao.UserDao;

import com.blog.system.Dto.CommentBean;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.UserBean;

public class DailyCommentServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

String param = request.getParameter("param");

if (param == null) {

param = "view";

}

if (param == "view" || param.equals("view")) {

getDailyComment(request, response);

} else if (param == "add" || param.equals("add")) {

String userid1 = request.getParameter("userid1");

int userid = Integer.parseInt(userid1);

String comment = request.getParameter("comment");

String pdid1 = request.getParameter("dailyid");

int pdid = Integer.parseInt(pdid1);

CommentBean cb = new CommentBean();

cb.setUserid(userid);

cb.setComment(comment);

cb.setPdid(pdid);

CommentDao cd = new CommentDao();

try {

cd.addComment(cb, 1);

} catch (Exception e) {

e.printStackTrace();

response.sendRedirect("/index.html");

}

response.sendRedirect(request.getContextPath()

+ "/use/dailycdo?dailyid=" + pdid);

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

public void getDailyComment(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

int page;

try {

if (request.getParameter("page") != null) {

page = Integer.parseInt(request.getParameter("page"));

} else {

page = 1;

}

int size = 2;

String dailyid = request.getParameter("dailyid");

List<CommentBean> ret = new ArrayList<CommentBean>();

CommentDao cd = new CommentDao();

ret = cd.getTypeComment(dailyid, 1, page, size);

long count = cd.getTypeCommentCount(dailyid, 1);

DailyDao dd = new DailyDao();

DailyBean db = dd.getSingleDaily(dailyid);

String userid = Integer.toString(db.getUserid());

UserDao ud = new UserDao();

UserBean ub = ud.getSingleUser(userid);

request.setAttribute("userinfo", ub);

request.setAttribute("comment", ret);

request.setAttribute("count", count);

request.setAttribute("page", page);

request.setAttribute("size", size);

request.setAttribute("singledaily", db);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/daily\_comment.jsp");

requestDispatcher.forward(request, response);

} catch (Exception e) {

e.printStackTrace();

}

}

}

//前台相册的Servlet

package com.blog.use.action;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.blog.system.Dao.\*;

import com.blog.system.Dto.\*;

public class PhotoalbumServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

String param = request.getParameter("param");

if (param == null) {

param = "view";

}

if (param == "view" || param.equalsIgnoreCase("view")) {

int page;

if (request.getParameter("page") != null) {

page = Integer.parseInt(request.getParameter("page"));

} else {

page = 1;

}

int size = 12;

String userid = request.getParameter("userid");

List<PhotoalbumBean> ret = new ArrayList<PhotoalbumBean>();

PhotoalbumDao pad = new PhotoalbumDao();

try {

ret = pad.getPhotoalbum(page, size, userid);

long count = pad.getPhotoalbumCount(userid);

UserDao ud = new UserDao();

UserBean ub = ud.getSingleUser(userid);

request.setAttribute("userinfo", ub);

request.setAttribute("userid", userid);

request.setAttribute("photoalbum", ret);

request.setAttribute("count", count);

request.setAttribute("page", page);

request.setAttribute("size", size);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/photo\_album\_list.jsp");

requestDispatcher.forward(request, response);

} catch (Exception e) {

e.printStackTrace();

}

} else if (param == "add" || param.equals("add")) {

String albumname = request.getParameter("albumname");

String albumdepict = request.getParameter("albumdepict");

//System.out.println(albumname);

PhotoalbumBean pab = new PhotoalbumBean();

String userid = request.getParameter("userid");

int userid1 = Integer.parseInt(userid);

pab.setUserid(userid1);

pab.setAlbumname(albumname);

pab.setAlbumdepict(albumdepict);

PhotoalbumDao pad = new PhotoalbumDao();

try {

pad.addPhotoalbum(pab);

response.sendRedirect(request.getContextPath()

+ "/use/photoalbumlistdo?userid=" + userid);

} catch (Exception e) {

e.printStackTrace();

response.sendRedirect("/index.html");

}

}else if (param == "preadd" || param.equals("preadd")) {

String userid = request.getParameter("userid");

UserDao ud = new UserDao();

UserBean ub = ud.getSingleUser(userid);

request.setAttribute("userinfo", ub);

request.setAttribute("userid", userid);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/addphotoalbum.jsp");

requestDispatcher.forward(request, response);

}else if (param == "del" || param.equals("del")) {

try {

String userid = request.getParameter("userid");

String albumid = request.getParameter("albumid");

PhotoalbumDao pad = new PhotoalbumDao();

pad.delPhotoalbum(albumid);

response.sendRedirect(request.getContextPath()

+ "/use/photoalbumlistdo?userid=" + userid);

} catch (Exception e) {

e.printStackTrace();

}

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

//前台评论的Servlet

package com.blog.use.action;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.blog.system.Dao.CommentDao;

import com.blog.system.Dao.DailyDao;

import com.blog.system.Dao.MessageDao;

import com.blog.system.Dao.UserDao;

import com.blog.system.Dto.CommentBean;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.MessageBean;

import com.blog.system.Dto.UserBean;

public class MessageServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doPost(request, response);

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

String param = request.getParameter("param");

if (param == null) {

param = "view";

}

if (param == "view" || param.equals("view")) {

getMessage(request, response);

} else if (param == "add" || param.equals("add")) {

String message = request.getParameter("message");

int blogid = Integer.parseInt(request.getParameter("blogid"));

int userid = Integer.parseInt(request.getParameter("userid"));

MessageBean mb = new MessageBean();

mb.setBlogid(blogid);

mb.setUserid(userid);

mb.setMessage(message);

MessageDao md = new MessageDao();

try {

md.addMessage(mb);

} catch (Exception e) {

e.printStackTrace();

response.sendRedirect("/index.html");

}

response.sendRedirect(request.getContextPath()

+ "/use/messagedo?blogid=" + blogid);

}

}

public void getMessage(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

int page;

try {

if (request.getParameter("page") != null) {

page = Integer.parseInt(request.getParameter("page"));

} else {

page = 1;

}

int size = 10;

String blogid = request.getParameter("blogid");

List<MessageBean> ret = new ArrayList<MessageBean>();

MessageDao md = new MessageDao();

ret = md.getMessage(page, size, blogid);

long count = md.getMessageCount(blogid);

UserDao ud = new UserDao();

UserBean ub = ud.getSingleUser(blogid);

request.setAttribute("userinfo", ub);

request.setAttribute("blogid", blogid);

request.setAttribute("message", ret);

request.setAttribute("count", count);

request.setAttribute("page", page);

request.setAttribute("size", size);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/message.jsp");

requestDispatcher.forward(request, response);

} catch (Exception e) {

e.printStackTrace();

}

}

}

//单个日志及其评论的Servelt

package com.blog.use.action;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.blog.system.Dao.CommentDao;

import com.blog.system.Dao.DailyDao;

import com.blog.system.Dao.UserDao;

import com.blog.system.Dto.CommentBean;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.UserBean;

public class DailyCommentServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

String param = request.getParameter("param");

if (param == null) {

param = "view";

}

if (param == "view" || param.equals("view")) {

getDailyComment(request, response);

} else if (param == "add" || param.equals("add")) {

String userid1 = request.getParameter("userid1");

int userid = Integer.parseInt(userid1);

String comment = request.getParameter("comment");

String pdid1 = request.getParameter("dailyid");

int pdid = Integer.parseInt(pdid1);

CommentBean cb = new CommentBean();

cb.setUserid(userid);

cb.setComment(comment);

cb.setPdid(pdid);

CommentDao cd = new CommentDao();

try {

cd.addComment(cb, 1);

} catch (Exception e) {

e.printStackTrace();

response.sendRedirect("/index.html");

}

response.sendRedirect(request.getContextPath()

+ "/use/dailycdo?dailyid=" + pdid);

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

public void getDailyComment(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

int page;

try {

if (request.getParameter("page") != null) {

page = Integer.parseInt(request.getParameter("page"));

} else {

page = 1;

}

int size = 2;

String dailyid = request.getParameter("dailyid");

List<CommentBean> ret = new ArrayList<CommentBean>();

CommentDao cd = new CommentDao();

ret = cd.getTypeComment(dailyid, 1, page, size);

long count = cd.getTypeCommentCount(dailyid, 1);

DailyDao dd = new DailyDao();

DailyBean db = dd.getSingleDaily(dailyid);

String userid = Integer.toString(db.getUserid());

UserDao ud = new UserDao();

UserBean ub = ud.getSingleUser(userid);

request.setAttribute("userinfo", ub);

request.setAttribute("comment", ret);

request.setAttribute("count", count);

request.setAttribute("page", page);

request.setAttribute("size", size);

request.setAttribute("singledaily", db);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/daily\_comment.jsp");

requestDispatcher.forward(request, response);

} catch (Exception e) {

e.printStackTrace();

}

}

}

//前台显示blog列表的Servlet

package com.blog.use.action;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.blog.system.Dao.DailyDao;

import com.blog.system.Dao.UserDao;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.UserBean;

public class BlogListServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doPost(request, response);

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

int page;

if (request.getParameter("page") != null) {

page = Integer.parseInt(request.getParameter("page"));

} else {

page = 1;

}

int size = 10;

List<UserBean> ret = new ArrayList<UserBean>();

UserDao ud = new UserDao();

ret = ud.getUser(page, size);

long count = ud.getUserCount();

request.setAttribute("user", ret);

request.setAttribute("count", count);

request.setAttribute("page", page);

request.setAttribute("size", size);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/blog\_list.jsp");

requestDispatcher.forward(request, response);

}

}

//前台显示博客主页的Servlet

package com.blog.use.action;

import java.io.IOException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.blog.system.Dao.DailyDao;

import com.blog.system.Dao.UserDao;

import com.blog.system.Dto.\*;

public class IndexServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doPost(request, response);

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

String userid = request.getParameter("userid");

UserDao ud = new UserDao();

UserBean ub = ud.getSingleUser(userid);

request.setAttribute("userinfo", ub);

DailyDao dd = new DailyDao();

DailyBean db = dd.getIndexDaily(userid);

request.setAttribute("indexdaily", db);

request.setAttribute("userid", userid);

RequestDispatcher requestDispatcher = request

.getRequestDispatcher("/use/index.jsp");

requestDispatcher.forward(request, response);

}

}

//日志的Dao

package com.blog.system.Dao;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import com.blog.util.db.\*;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.UserBean;

public class DailyDao {

private Connection con;

DailyBean db = new DailyBean();

public void setDaily(DailyBean db) {

this.db = db;

}

public DailyBean getIndexDaily(String userid1) {

try {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

int userid = Integer.parseInt(userid1);

ResultSet result = stm

.executeQuery("select \* from daily where userid = "

+ userid + " order by dailyid desc limit 1 ");

while (result.next()) {

int dailyid = result.getInt("dailyid");

String dailyname = result.getString("dailyname");

String daily = result.getString("daily");

java.sql.Date postingdate = result.getDate("postingdate");

java.sql.Date modifytime = result.getDate("modifytime");

db.setUserid(userid);

db.setDailyid(dailyid);

db.setDailyname(dailyname);

db.setDaily(daily);

db.setPostingdate(postingdate);

db.setModifytime(modifytime);

}

result.close();

stm.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return db;

}

public List<DailyBean> getDaily(int page, int size, String userid1)

throws Exception {

List<DailyBean> ret = new ArrayList<DailyBean>();

int begin = (page - 1) \* size;

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

ResultSet result = stm.executeQuery("select \* from daily where userid="

+ userid1 + " order by dailyid limit " + begin + "," + size);

while (result.next()) {

int userid = result.getInt("userid");

int dailyid = result.getInt("dailyid");

String dailyname = result.getString("dailyname");

java.sql.Date postingdate = result.getDate("postingdate");

java.sql.Date modifytime = result.getDate("modifytime");

DailyBean db = new DailyBean();

db.setUserid(userid);

db.setDailyid(dailyid);

db.setDailyname(dailyname);

db.setPostingdate(postingdate);

db.setModifytime(modifytime);

ret.add(db);

}

result.close();

stm.close();

con.close();

return ret;

}

public List<DailyBean> getSystemDaily(int page, int size) throws Exception {

List<DailyBean> ret = new ArrayList<DailyBean>();

int begin = (page - 1) \* size;

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select \* from daily order by dailyid desc limit "

+ begin + "," + size);

while (result.next()) {

int userid = result.getInt("userid");

int dailyid = result.getInt("dailyid");

String dailyname = result.getString("dailyname");

java.sql.Date postingdate = result.getDate("postingdate");

java.sql.Date modifytime = result.getDate("modifytime");

DailyBean db = new DailyBean();

db.setUserid(userid);

db.setDailyid(dailyid);

db.setDailyname(dailyname);

db.setPostingdate(postingdate);

db.setModifytime(modifytime);

ret.add(db);

}

result.close();

stm.close();

con.close();

return ret;

}

public Long getDailyCount() {

long l = 0;

con = GetDbCon.getDbConn();

try {

Statement stm = con.createStatement();

ResultSet result = stm.executeQuery("select count(\*) from daily");

if (result.next()) {

l = result.getLong(1);

}

} catch (SQLException e) {

e.printStackTrace();

}

return l;

}

public Long getUseDailyCount(String userid) {

long l = 0;

con = GetDbCon.getDbConn();

try {

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select count(\*) from daily where userid ="

+ userid);

if (result.next()) {

l = result.getLong(1);

}

} catch (SQLException e) {

e.printStackTrace();

}

return l;

}

public void addDaily(DailyBean db) {

try {

con = GetDbCon.getDbConn();

PreparedStatement stm = con

.prepareStatement("insert into daily(dailyname,daily,postingdate,userid) values(?,?,?,?)");

stm.setString(1, db.getDailyname());

stm.setString(2, db.getDaily());

stm.setDate(3, new java.sql.Date(new java.util.Date().getTime()));

stm.setInt(4, db.getUserid());

stm.execute();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public void updateDaily(DailyBean db) {

try {

con = GetDbCon.getDbConn();

PreparedStatement stm = con

.prepareStatement("update daily set dailyname=?,daily=?,modifytime=? where dailyid=?");

stm.setString(1, db.getDailyname());

stm.setString(2, db.getDaily());

stm.setDate(3, new java.sql.Date(new java.util.Date().getTime()));

stm.setInt(4, db.getDailyid());

stm.execute();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public void delDaily(String dailyid) throws SQLException {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

stm.execute("DELETE FROM daily where dailyid ='" + dailyid + "'");

}

public DailyBean getSingleDaily(String dailyid) {

DailyBean db = new DailyBean();

try {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

int dailyid1 = Integer.parseInt(dailyid);

ResultSet result = stm

.executeQuery("select \* from daily where dailyid = "

+ dailyid1);

while (result.next()) {

int userid = result.getInt("userid");

String dailyname = result.getString("dailyname");

String daily = result.getString("daily");

java.sql.Date postingdate = result.getDate("postingdate");

java.sql.Date modifytime = result.getDate("modifytime");

db.setUserid(userid);

db.setDailyid(dailyid1);

db.setDailyname(dailyname);

db.setDaily(daily);

db.setPostingdate(postingdate);

db.setModifytime(modifytime);

}

result.close();

stm.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return db;

}

}

//相册的Dao

package com.blog.system.Dao;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import com.blog.system.Dto.\*;

import com.blog.util.db.GetDbCon;

public class PhotoalbumDao {

private Connection con;

PhotoalbumBean pab = new PhotoalbumBean();

public void setPhotoalbum(PhotoalbumBean pab) {

this.pab = pab;

}

public List<PhotoalbumBean> getPhotoalbum(int page, int size, String userid1)

throws Exception {

List<PhotoalbumBean> ret = new ArrayList<PhotoalbumBean>();

int begin = (page - 1) \* size;

try {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select \* from photoalbum where userid='"

+ userid1 + "' limit " + begin + "," + size);

while (result.next()) {

int albumid = result.getInt("albumid");

int userid = result.getInt("userid");

String albumname = result.getString("albumname");

String albumdepict = result.getString("albumdepict");

java.sql.Date createtime = result.getDate("createtime");

PhotoalbumBean pab = new PhotoalbumBean();

pab.setAlbumid(albumid);

pab.setAlbumname(albumname);

pab.setAlbumdepict(albumdepict);

pab.setCreatetime(createtime);

pab.setUserid(userid);

ret.add(pab);

}

result.close();

stm.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return ret;

}

public Long getPhotoalbumCount(String userid) {

long l = 0;

con = GetDbCon.getDbConn();

try {

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select count(\*) from photoalbum where userid = "

+ userid);

if (result.next()) {

l = result.getLong(1);

}

} catch (SQLException e) {

e.printStackTrace();

}

return l;

}

public void delPhotoalbum(String albumid) throws SQLException {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

stm.execute("DELETE FROM photoalbum where albumid ='" + albumid + "'");

}

public void addPhotoalbum(PhotoalbumBean pab) {

try {

con = GetDbCon.getDbConn();

PreparedStatement stm = con

.prepareStatement("insert into photoalbum(albumname,albumdepict,createtime,userid) values(?,?,NOW(),?)");

stm.setString(1, pab.getAlbumname());

stm.setString(2, pab.getAlbumdepict());

//stm.setDate(3, new java.sql.Date(new java.util.Date().getTime()));

stm.setInt(3, pab.getUserid());

stm.execute();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public String getAlbumname(String albumid) {

String albumname = null;

try {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select \* from photoalbum where albumid="

+ albumid);

while (result.next()) {

albumname = result.getString("albumname");

PhotoalbumBean pab = new PhotoalbumBean();

pab.setAlbumname(albumname);

}

result.close();

stm.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return albumname;

}

}

//图片的DAO

package com.blog.system.Dao;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import com.blog.system.Dto.DailyBean;

import com.blog.system.Dto.PhotoBean;

import com.blog.util.db.GetDbCon;

public class PhotoDao {

private Connection con;

PhotoBean pb = new PhotoBean();

public void setPhoto(PhotoBean pb) {

this.pb = pb;

}

public List<PhotoBean> getPhoto(int page, int size, String userid1,String albumid)

throws Exception {

List<PhotoBean> ret = new ArrayList<PhotoBean>();

int begin = (page - 1) \* size;

try {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select \* from photoinfo where userid='"

+ userid1 + "' and albumid='"+albumid+"'limit " + begin + "," + size);

while (result.next()) {

int photoid = result.getInt("photoid");

int userid = result.getInt("userid");

int albumid1 = result.getInt("albumid");

String photoname = result.getString("photoname");

String photodepict = result.getString("photodepict");

String photoaddress = result.getString("photoaddress");

java.sql.Date uploadtime = result.getDate("uploadtime");

PhotoBean pb = new PhotoBean();

pb.setAlbumid(albumid1);

pb.setPhotoid(photoid);

pb.setUserid(userid);

pb.setPhotoname(photoname);

pb.setPhotoaddress(photoaddress);

pb.setPhotodepict(photodepict);

pb.setUploadtime(uploadtime);

ret.add(pb);

}

result.close();

stm.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return ret;

}

public List<PhotoBean> getSystemPhoto(int page, int size)

throws Exception {

List<PhotoBean> ret = new ArrayList<PhotoBean>();

int begin = (page - 1) \* size;

try {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select \* from photoinfo limit " + begin + "," + size);

while (result.next()) {

int photoid = result.getInt("photoid");

int userid = result.getInt("userid");

String photoname = result.getString("photoname");

String photodepict = result.getString("photodepict");

String photoaddress = result.getString("photoaddress");

java.sql.Date uploadtime = result.getDate("uploadtime");

PhotoBean pb = new PhotoBean();

pb.setPhotoid(photoid);

pb.setUserid(userid);

pb.setPhotoname(photoname);

pb.setPhotoaddress(photoaddress);

pb.setPhotodepict(photodepict);

pb.setUploadtime(uploadtime);

ret.add(pb);

}

result.close();

stm.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return ret;

}

public Long getPhotoCount() {

long l = 0;

con = GetDbCon.getDbConn();

try {

Statement stm = con.createStatement();

ResultSet result = stm

.executeQuery("select count(\*) from photoinfo");

if (result.next()) {

l = result.getLong(1);

}

} catch (SQLException e) {

e.printStackTrace();

}

return l;

}

public void delPhoto(String photoid) throws SQLException {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

stm.execute("DELETE FROM photoinfo where photoid ='" + photoid + "'");

}

public PhotoBean getSinglePhoto(String photoid) {

PhotoBean pb = new PhotoBean();

try {

con = GetDbCon.getDbConn();

Statement stm = con.createStatement();

int photoid1 = Integer.parseInt(photoid);

ResultSet result = stm

.executeQuery("select \* from photoinfo where photoid = "

+ photoid1);

while (result.next()) {

int userid = result.getInt("userid");

String photoname = result.getString("photoname");

String photoaddress = result.getString("photoaddress");

String photodepict = result.getString("photodepict");

java.sql.Date uploadtime = result.getDate("uploadtime");

pb.setUserid(userid);

pb.setPhotoid(photoid1);

pb.setPhotoname(photoname);

pb.setPhotoaddress(photoaddress);

pb.setPhotodepict(photodepict);

pb.setUploadtime(uploadtime);

}

result.close();

stm.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return pb;

}

public void addPhoto(PhotoBean pb) {

try {

con = GetDbCon.getDbConn();

PreparedStatement stm = con

.prepareStatement("insert into photoinfo(photoname,photoaddress,photodepict,uploadtime,userid,albumid) values(?,?,?,?,?,?)");

stm.setString(1, pb.getPhotoname());

stm.setString(2, pb.getPhotoaddress());

stm.setString(3, pb.getPhotodepict());

stm.setDate(4, new java.sql.Date(new java.util.Date().getTime()));

stm.setInt(5, pb.getUserid());

stm.setInt(6, pb.getAlbumid());

stm.execute();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

验证登陆的JSP页面

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

<%@ page import="java.sql.\*"%>

<%

// 接收请求的内容

String username = request.getParameter("username");

String password = request.getParameter("password");

Connection con = null;

String CLASSFORNAME = "com.mysql.jdbc.Driver";

String SERVANDDB = "jdbc:mysql://localhost:3306/blog";

String USER = "root";

String PWD = "chg880918";

String url = SERVANDDB + "?user=" + USER + "&password=" + PWD;

Class.forName(CLASSFORNAME).newInstance();

con = DriverManager.getConnection(url);

Statement stmt = con.createStatement();

ResultSet rs = stmt

.executeQuery("select \* from userinfo where username='"

+ username + "' and password = '" + password+"'");

if (rs.next()) {

int userid = rs.getInt("userid");

session.setAttribute("viewuserid",userid);

System.out.println(userid);

response.sendRedirect("bloglistdo");

} else {

//response.sendRedirect("login\_fail.jsp");

}

rs.close();

stmt.close();

con.close();

%>

动态验证输入是否正确的JS

var flag = [0, 0, 0];

function checkUsername(username){

var reId = /([\_\a-zA-Z0-9]){4,19}$/;

var b\_id = reId.test(username);

if (!b\_id) {

document.getElementById("userName\_note"). innerHTML = "<font color=red>未输入用户名或格式错误</font>";

flag[0] = 0;

} else{

document.getElementById("userName\_note"). innerHTML = "<font color=#ee9955>输入合法</font>";

flag[0] = 1;

}

}

function checkPassword(password){

var pwId = /([\uFE30-\uFFA0]|[\_\a-zA-Z0-9]){6,19}$/;

var p\_id = pwId.test(password);

if (!p\_id) {

document.getElementById("passWord\_note"). innerHTML = "<font color=red>检查密码格式</font>";

flag[1] = 0;

} else{

document.getElementById("passWord\_note"). innerHTML = "<font color=#ee9955>密码合法</font>";

flag[1] = 1;

}

}

function checkrepassword(repassword){

var password=document.getElementById("password1").value;

var secondPassword=document.getElementById("password2").value;

if(password!=secondPassword)

{

document.getElementById("SecondPassword\_note"). innerHTML = "<font color=red>二次密码输入不一致</font>";

flag[2] = 0;

} else

{

document.getElementById("SecondPassword\_note"). innerHTML = "<font color=#ee9955>密码合法</font>";

flag[2] = 1;

}

}

function registSubmit(){

if(flag[0]==0){

alert ("请输入正确用户名")

return false;

}else if(flag[1]==0){

alert ("请检查两次密码是否输入正确")

return false;

}else if(flag[2]==0){

alert ("请检查两次密码是否输入正确")

return false;

}

}