`目 录

第一章 绪论

1.1 系统的目的和意义

系统功能：

本系统是用 JSP 实现的一个完全基于浏览器的博客系统
任何注册个人博客的人都拥有以下功能

1. 登录博客系统的后台管理平台。
2. 更改博客账号的基本信息。包括密码等
3. 撰写日志。
4. 日志管理。
5. 评论管理。
6. 留言管理。

博客访问者具有以下功能

1. 浏览博客系统中的日志、文章
2. 发表评论及留言。
3. 下载文件、软件、图片等

系统意义：

随着 Blog 人数的增加 Blog 作为一种新的生活方式、新的工作方式和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。它使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得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。

本系统采用 MVC 模式设计实现了一个简单的博客系统。MVC 模式极大地提高了系统的灵活性、复用性、开发效率、适应性和可维护性 充分发挥了 JSP、Servlet 等 J2EE 组件的特点，从而使更多的人们通过文字、图片、声音、视频、无线等尽情展示自我、分享感受、参与交流美好你我生活。Show you, share me, 人人都可以博客 人人都需要博客

第二章 个人博客需求分析

2.1 引言

需求背景：

通过写博客、记录自己的想法可以帮助实现如下这些目的
1、记录生活　在自己的每一天上留下思考的划痕。
2、整理思路　固化知识　获得更多更好的想法。
3、分享　　将自己的想法、经验与人分享。
4、交流、提高 通过交流产生更多的思维火花　相互提高。
5、交友　　互相鼓励　一路同行。

如果说博客网站是一个舞台，那么所有的博客都是其中的舞者，把自己有价值的真实的一面展示在网络世界中相互交流沟通，如果博客仅仅为了写日志那么博客是一面镜子，镜子中展现的是真实的自己，如果博客是一个自由媒体可以把自身感觉有价值的信息通过博客这个工具发布与网友共享。

使用的开发技术如下：

1、jsp 的基础：servlet 技术；
2、jsp+JavaBean 实现业务逻辑和页面显示；
3、JDBC 进行数据库操作，实现信息的读写；
4、利用 Model2 的 MVC 设计模式减少代码冗余，实现代码的可重用性；

2.2 需求分析

主要内容：

本系统是用 JSP 实现的一个完全基于浏览器的博客系统。任何注册个人博客的人都拥有登录博客系统的后台管理平台、更改博客账号的基本信息、撰写日志、日志管理、评论管理、留言管理。 博客访问者可以浏览博客系统中的日志、留言、评论。

具体功能：

1、系统包含用户注册与登录的功能模块，登录时需要提供验证码识别功能。
2、服务器端提供识别是否为登录用户的安全监测 Filter，用于过滤绕过登录模块直接 访问的非正常访问。
3、文章管理实现对数据库的增、删、改、查的操作；对数据库的访问代码封装到专门的 JavaBean 中；实现专门用于内容分页显示的功能模块；
4、需要实现文件的上传功能。
5、通过 HttpSession 对象实现用户会话管理，实现统计当前在线用户人数；
6、整个系统采用 UTF-8 编码，数据库也采用 UTF-8 格式存储，避免乱码问题。
7、客户端界面要求简洁美观。既要体现系统功能，也需要符合一般网站的美工与布局 要求。推荐使用 Bootstrap 或 JQuery 开发库实现。

2.3 功能需求分析

由于是个人博客系统，因此本系统用户首先需要登录该系统得到权限后才能对系统进行操作，未注册用户可以直接进入注册模块进行注册。登录系统后确定权限只有管理员权限 才能对系统进行维护。普通用户可以对个人主页进行管理管理的板块包括：日记、相册、留言。

2.4 数据流图需求分析

系统运行过程中，主要涉及到两部分的数据流向，一部分是用户在系统后台管理界面上发出的对数据库操作的数据流，另一部分是用户在应用前台访问文章和留言时所产生的数据流。

2.5 数据库需求分析

为了把用户的数据要求清晰明确地表达出来 通常要建立一个概念性的数据模型。概念性数据模型是一种面向问题的数据模型 是按照用户的观点来对数据和信息建模。描述了从用户角度看到的数据，反映了用户的现实环境。最常用的表示概念性数据模型的方法是实体—联系方法。这种方法用 ER 图描述现实世界中的实体而不涉及这些实体在系统中的实现方法。用这种方法表示的概念性数据模型又称为 ER 模型。ER 模型中包含“实体” “联系”和“属性”。
博客信息管理系统涉及的实体包括：

1.用户:用户 ID、用户名称、用户密码、E-mail、电话

2.日志：日志编号、日志标题、日志内容、发表日期、作者、类别、文章路径、浏览数、状态。

3.留言: 留言编号、留言内容、留言人联系方式。

4.文件：文件编号、文件名、文件路径、文件的备注。

第三章 个人博客系统结构分析与设计

3.1 引言

根据系统的需求分析，博客管理系统是一个集网络、数据库于一体的综合系统，因此系统在总体设计时应遵循以下原则：

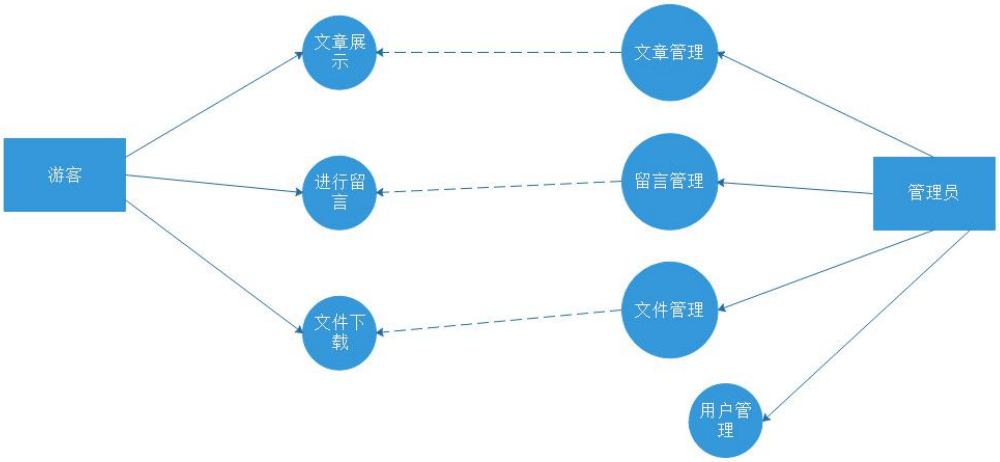
(1) 稳定性、实用性、良好的可扩充性和安全性，提供良好的人机界面，界面简洁，操作简单。

(2) 系统的功能设计完善，能够有效解决用户使用中出现的问题，满足各个方面的使用需要。

(3) 数据库结构设计合理，字段属性要准确，字段长度要满足实际需要。

(4) 系统流程合理，能够符合博客管理及使用的基本操作流程。

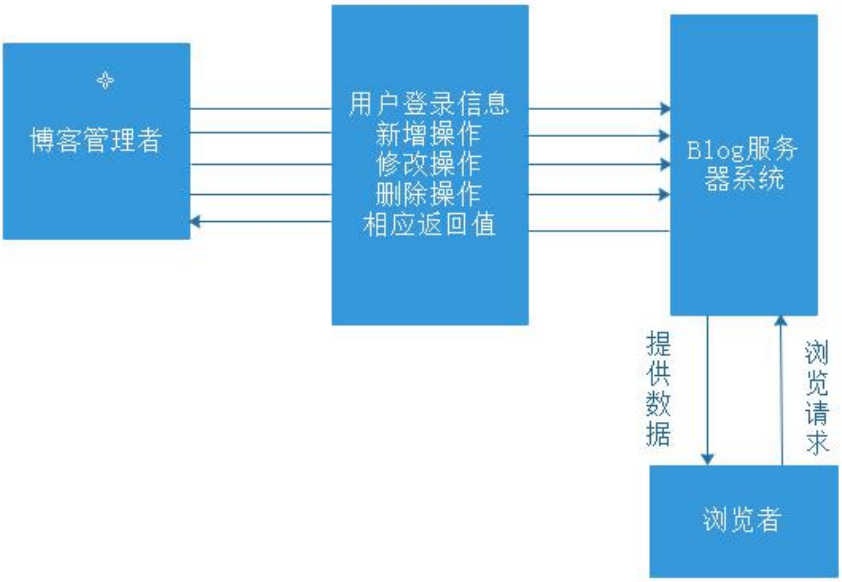
用例图：



3.2 系统结构图

博客系统中，主要是注册用户管理文章、留言、相册、好友等数据，数据的类型决定了程序对数据的处理方式也就是算法， 因此，数据是实现分析的起点。现通过数据流图的方式分析系统中的数据的流动和处理。

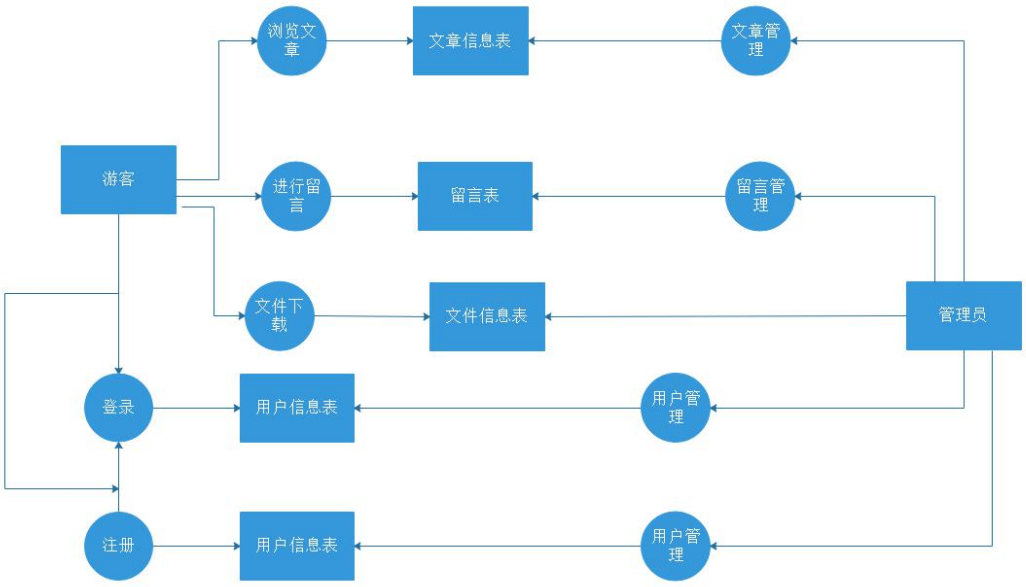
博客系统流图：



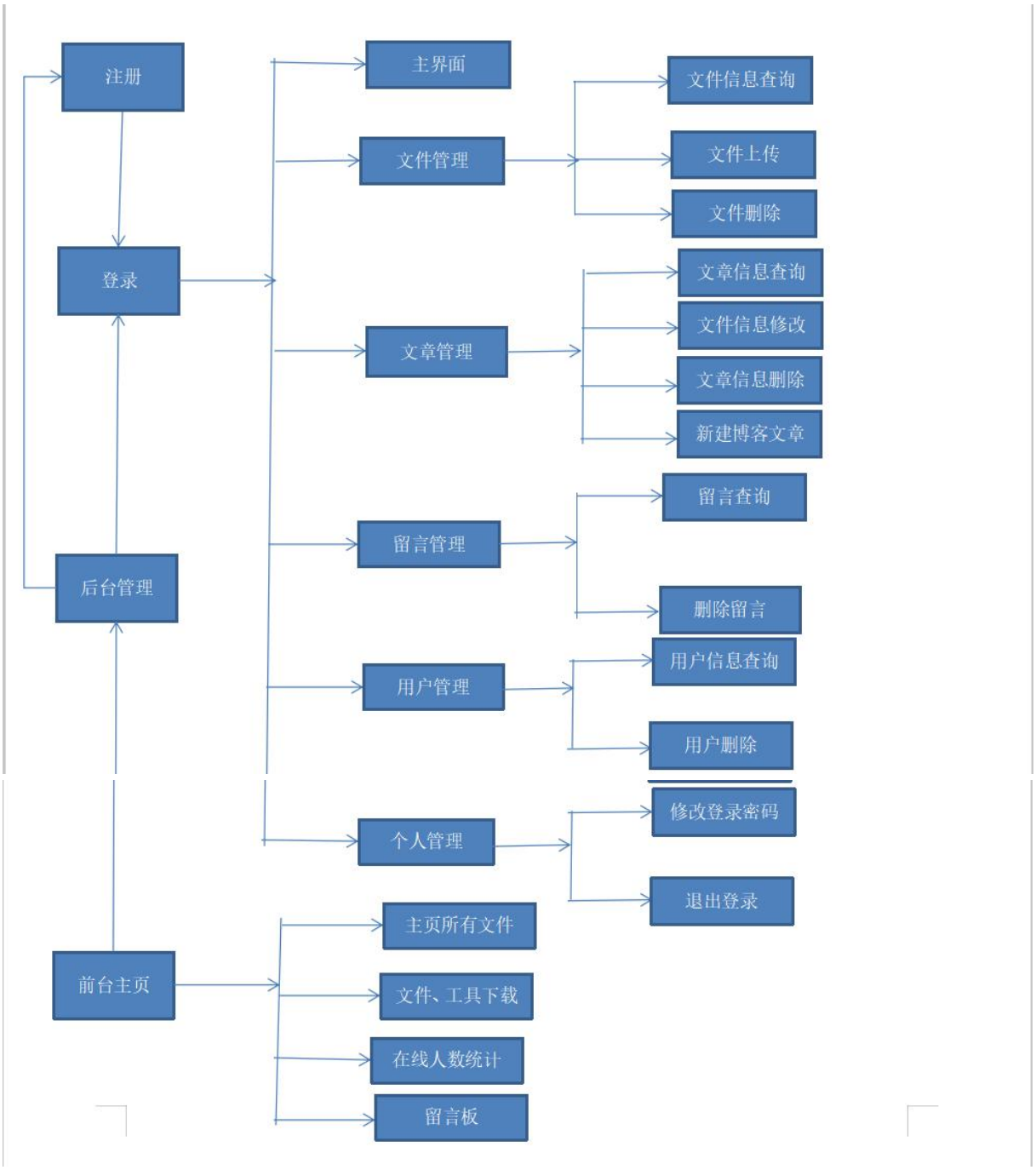
注册用户可以注册、登录本系统，对文章、文件、留言等进行管理，其中包括对各个模块的信息的增、删、改、查操作。

网友可以访问博主的博客，浏览博主的文章、相册、好友等信息，还可以给博主留言。

其具体业务逻辑流程如图：



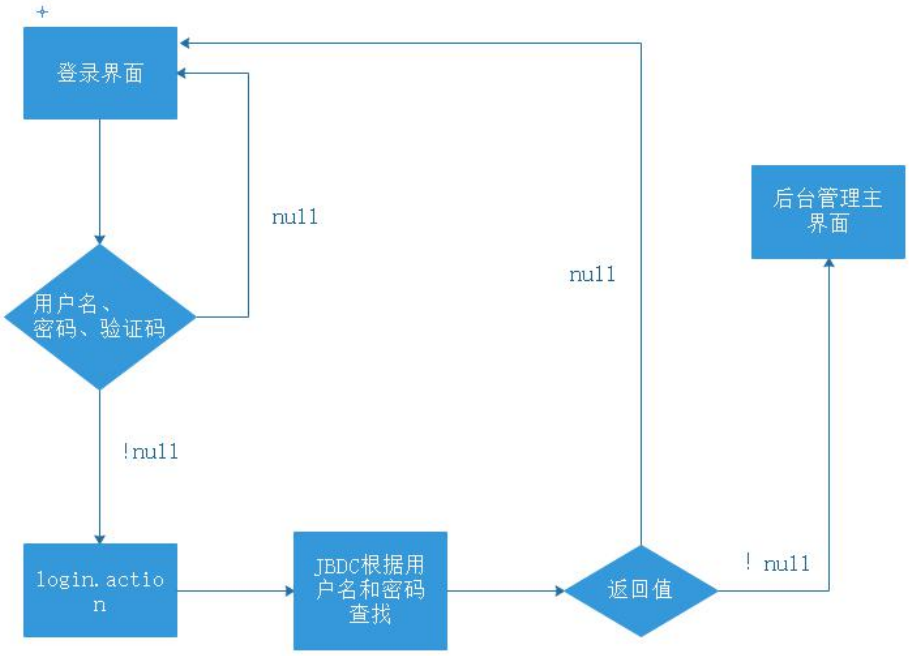
根据功能划分出了整个系统的功能图：



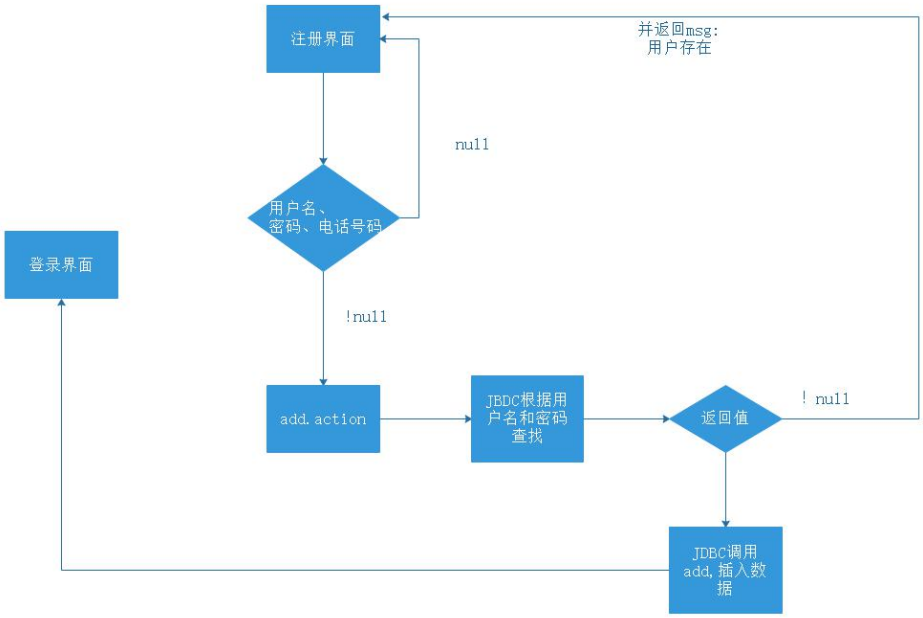
3.3 登录注册模块

登录：已存在用户可直接登录，提供验证码以及校验输入是否输入。首先将用户输入的名字和密码传入调用的 servlet 方法里，servlet 接收然后在将之作为参数调用 JDBC，对改用户名和密码进行查询，根据查询结果是否为空，不为空登陆成功！

登陆实现具体流程：



注册：用户可以在注册页面进行注册，提供用户检验是否已注册功能。与登录相似，但在前面会实现一个简单验证该用户是否已注册，具体流程如下：

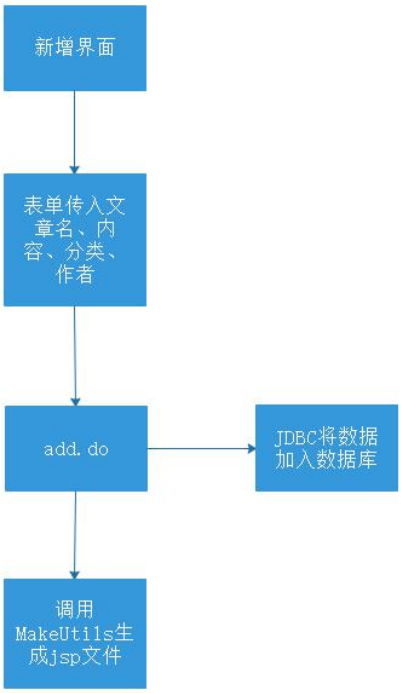


3.4 文章管理模块

3.4.1 新增文章：

对于自己要新建博客文章，则需要准备一个 jsp 模板，选择分类，输入文章标题，还有文章内容。新建时利用 MakeUtils 将文章内容写入模板，生成新的 jsp，并把其放在相应的文件夹下，然后将文章信息保存到数据库中

具体流程如下：



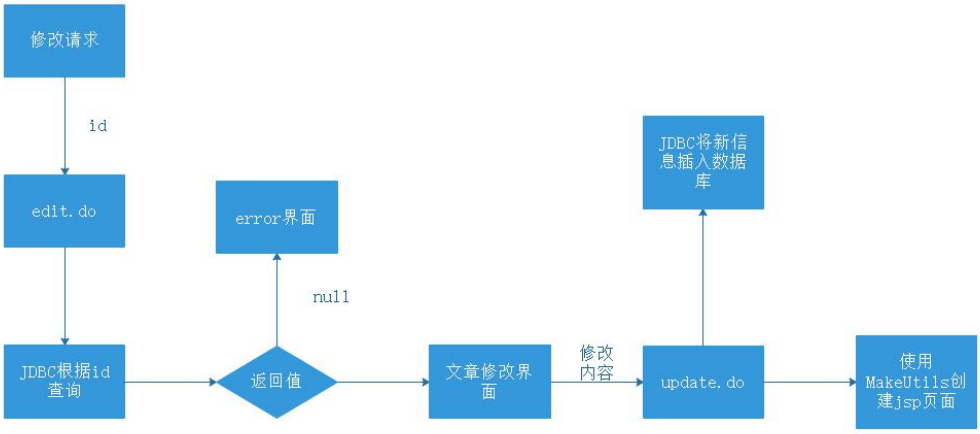
3.4.1 文章删除：

流程较简单，主要是根据传入的 id 值作为参数，利用 JDBC 删除数据库中 id 所对应的表项。

3.4.1 文章修改：

这里需要注意要先判断文章是否存在，在进行修改，因此要先根据 id 的值作为参数，利用 JDBC 进行数据库的查询，判断其是否存在，存在，就跳转到修改页面，然后对文章内容进行修改！

具体流程如下：

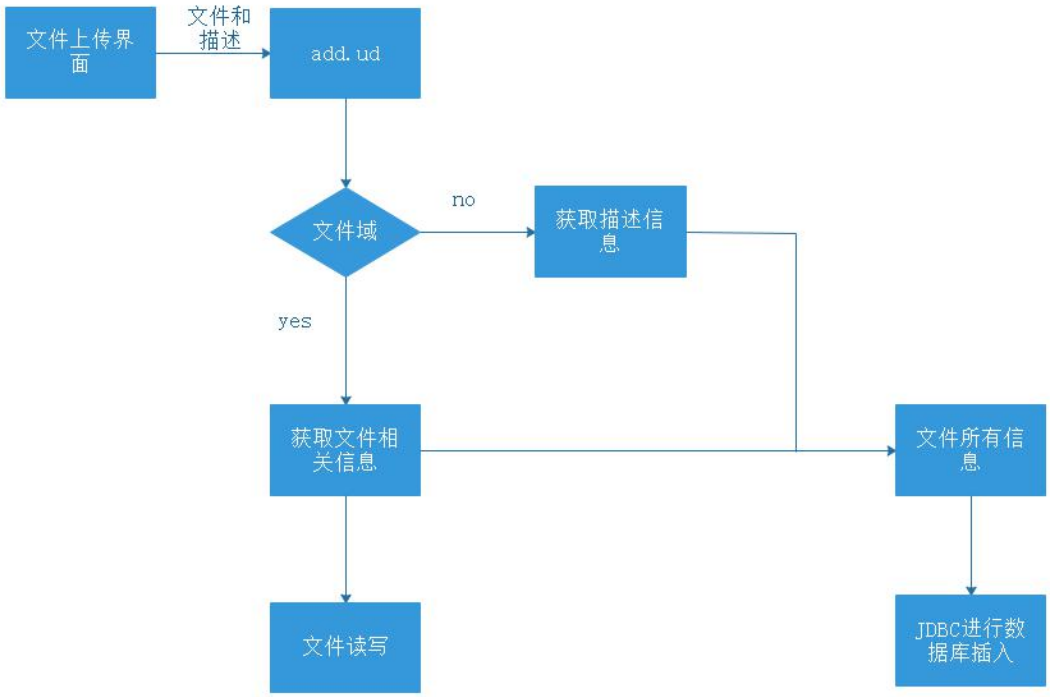


3.5 文件管理模块

3.5.1 文件上传：

利用 fileupload 和 io 组件实现文件上传，利用官方文档，创建 factory，使用 factory 设置上传文件的大小还有最大空间等等。然后接收所有上传请求的二进制流。在遍历该二进制流，判断其是文件域还是表单域，可以从表单域中设置存取的路径和文件的文件名，将之保存到数据库中，在用 bytes[]将文件进行读写！

具体流程如下：



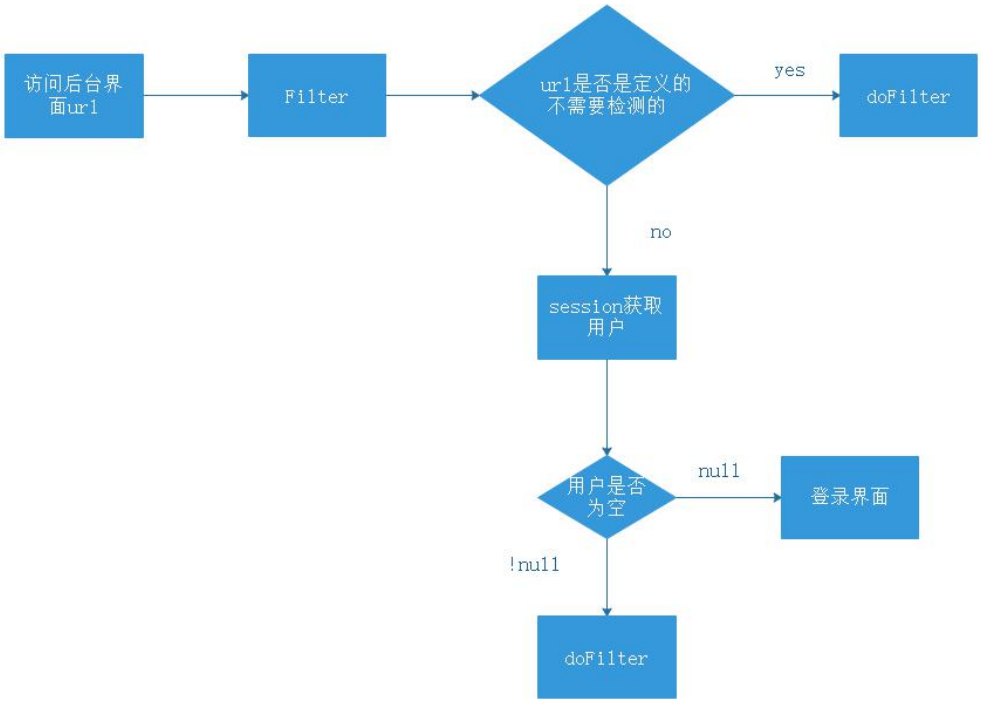
3.5.1 文件下载：

先利用 JDBC 将要下载文件的目录，文件名等等信息全部获取，然后再利用输出流进行下载。

3.6 过滤器模块

利用 Filter,对界面进行过滤，当用户未直接访问后台页面，则会先调用 Filter 检测 session 中是否存在用户，若存在，则表示已登录，若不存在，则拦截用户，表示未登录。

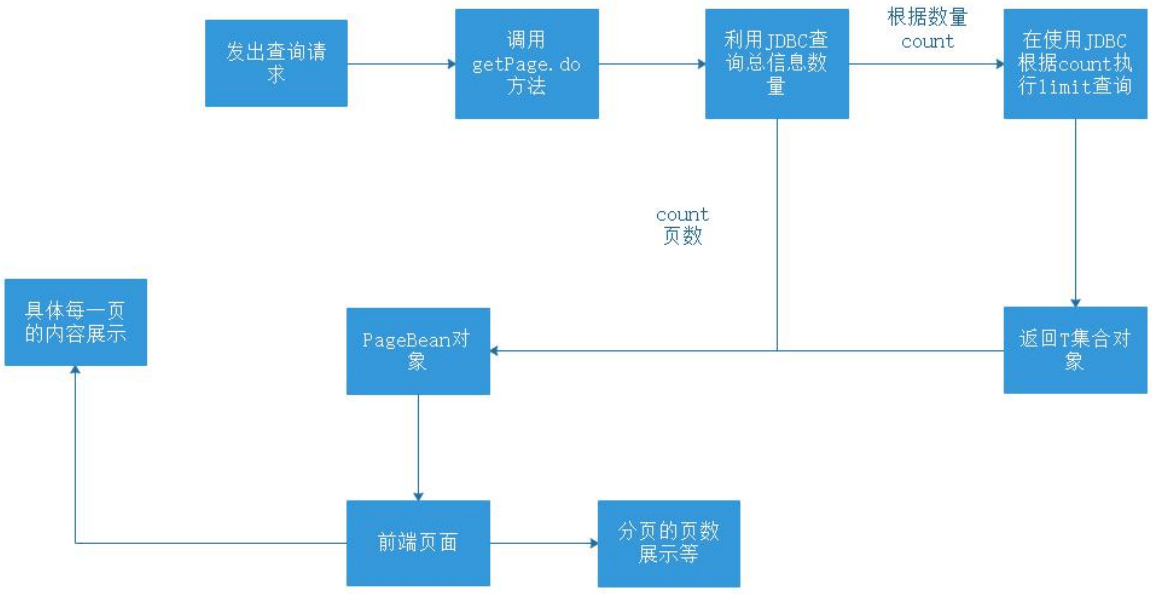
具体流程如下：



3.7 分页模块

，分页的实现，首先要写一个 Page 的 JavaBean,里面包含页数，当前页码，存放一个泛型的 List 集合。每一次查询访问时，都会调用 servlet 中的相应方法，然后用 JDBC 实现信息的存储，最后将信息放给前端展示，这里难点时数据库的访问和在前端展示页面。数据库的访问需要先查询出所有信息的数量，在利用 limit 的 sql 语句进行查询。

具体流程图如下：



第四章 关键技术研究

4.1 实现技术路线

整个系统采用 JSP 的 Model2 实现基于 MVC 设计模式的软件架构。通过 JSP+Servlet+JavaBean 的模式实现系统基础架构。提高了系统的可复用性。简单实现了业务逻辑和数据持久层分离。

平台采用 apache 旗下的 Tomcat 容器。

客户端主要使用 jsp+js 表现界面，JSP 即 Java Server Pages，是基于 Java Servlet 以及整个 Java 体系的 Web 开发技术。JSP 已经成为开发 Web 动态网页重要、 快速和有效的工具， 是全新的网络服务器编程环境。JSP 充分利用了 Java 的强大功能， 是一种优秀的服务器端技术。随着 Java 技术的日益成熟和流行，JSP 在网络编程中也变得越来越重要。JSP 基于强大的 Java 语言，具有极强的扩展能力，良好的缩收性，与平台无关的开发特性，成为构建动态网站的主流技术之一，JSP 有着其他技术所不具备的优势。其优势在于：可以将内容的生成和显示进行分离、生成可重用的组件、采用标识简化页面开发。

而服务端采用 java 语言为基础的 servlet+JDBC 实现请求的接受转发以及对数据库的访问。

数据库采用 MySQL，其好处如下：

(1) MySQL 是一种数据库管理系统。计算机是处理大量数据的理想工具，因此数据库管理系统在计算方面扮演着关键的中心角色，或是作为独立的实用工具，或是作为其他应用程序的组成部分。

(2) MySQL 是一种关联数据库管理系统。关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大的仓库内。这样就增加了速度并提高了灵活性。

(3) MySQL 软件是一种开放源码软件。

(4) MySQL 数据库服务器具有快速、可靠和易于使用的特点。 MySQL 服务器有一套实用的特性集合，这些特性是通过与我们用户的密切合作而开发的。在我们的基准测试主页上，给出了 MySQL 服务器和其他数据库管理器的比较结果。

(5) MySQL 服务器工作在客户端 /服务器模式下，或嵌入式系统中。 MySQL 数据库软件是一种客户端 /服务器系统，由支持不同后端的 1 个多线程 SQL 服务器，数种不同的客户端程序和库，众多管理工具和广泛的应用编程接口 API 组成。

而整个项目实现的技术路线则是使用 jsp+servlet，基于其操作易懂，易于掌握原理！赢因此适合小项目，个人小网站上手实验！

4.2 关键技术研究

由于本项目采用的是 jsp+servlet 技术，其中最核心的是 servlet 相关的技术，而在数据持久层则是采用的 JDBC 进行数据库操作，其核心技术就是 JDBC。因此关键技术就是 servlet 和 JDBC

就 servlet 而言，其具有可移植性(Portability)：

Servlet 皆是利用 Java 语言来开发的，因此，延续 Java 在跨平台上的表现，不论 Server 的操作系统是 Windows、Solaris、Linux、HP-UX、FreeBSD、Compaq Tru 64、AIX 等等，都能够将我们所写好的 Servlet 程序放在这些操作系统上执行。

而在本项目中，servlet 主要是作为前端参数的接收，然后将之传递给 JDBC 的实现方法，实现的难点主要有对多个请求访问一个 servlet 的实现以及文件上传的使用。通过网上资料，对个请求访问使用反射进行解决，而文件上传则使用 fileupload 和 io 组件以及官方文档进行实现。

JDBC 用来进行数据库访问，开始都是利用 DriverManage 加载驱动、用户名等等进行数据库连接，逐步实现，代码重复高，代码量大，效率低，为提高效率采用数据库连接池，引入数据源，提高数据库连接操作的效率。使用过程中又发现很多方法都是类似的，代码的相似程度高，因此可以写一个通用的方法类 DAO。里面包含查找一个 T 对象以及 T 对象的集合，同时可以根据参数的值查找一个值等等。通过网上资料查找，以及自己对 java 语言的了解，利用可变参数进行参数传值，利用 dbutils 工具进行数据库访问，具体实现该方法。

4.2.1 JSP 技术

jsp 简介：

JSP 即 Java Server Pages，是基于 Java Servlet 以及整个 Java 体系的 Web 开发技术。JSP 已经成为开发 Web 动态网页重要、 快速和有效的工具， 是全新的网络服务器编程环境。JSP 充分利用了 Java 的强大功能， 是一种优秀的服务器端技术。随着 Java 技术的日益成熟和流行，JSP 在网络编程中也变得越来越重要。JSP 基于强大的 Java 语言，具有极强的扩展能力，良好的缩收性，与平台无关的开发特性，成为构建动态网站的主流技术之一，JSP 有着其他技术所不具备的优势。其优势在于：可以将内容的生成和显示进行分离、生成可重用的组件、采用标识简化页面开发。

Jsp 使用： jsp 主要用于接收后端 servlet 传递的值，并将之显示，常见的查询界面的展示：

```
<table class="result-tab" width="100%">
            <tr>
                <th class="tc" width="5%"><input class="allChoose" name="article" type="checkbox"></th>

                <th>ID</th>
                <th>Title</th>
                <th>status</th>
                <th>click</th>
                <th>Publisher</th>
                <th>UpdateTime</th>
                <th>flagstr</th>
                <th>Operation</th>
            </tr>
           <c:forEach items="${pbList.beanList }" var="article">
            <tr>

              <th class="tc" width="5%"> <input class="allChoose" name="ace" type="checkbox" value=""></th>
               <td>${article.ID }</td>
               <td><a target="_blank" href="download.do?titlefile=${article.title }" >${article.title }</a></td>
               <td>${article.status }</td>
               <td>${article.click }</td>
               <td>${article.publisher }</td>
               <td>${article.updateTime }</td>
               <td>${article.flagstr }</td>
               <td>
                   <a class="link-update" href="${pageContext.request.contextPath}/edit.do?id=${article.ID }">修改</a>
                   <a class="link-del" href="${pageContext.request.contextPath}/delete.do?ID=${article.ID }">删除</a>
               </td>
            </tr>
           </c:forEach>
        </table>
```

这里就是将文章查询出的所有信息在前端 jsp 页面用 el 表达式将之取出。

4.2.2 servlet 技术

Servlet 介绍：

Servlet（Server Applet）是 Java Servlet 的简称，称为小服务程序或服务连接器，用 Java 编写的服务器端程序，具有独立于平台和协议的特性，主要功能在于交互式地浏览和生成数据，生成动态 Web 内容。狭义的 Servlet 是指 Java 语言实现的一个接口，广义的 Servlet 是指任何实现了这个 Servlet 接口的类，一般情况下，人们将 Servlet 理解为后者。Servlet 运行于支持 Java 的应用服务器中。从原理上讲，Servlet 可以响应任何类型的请求，但绝大多数情况下 Servlet 只用来扩展基于 HTTP 协议的 Web 服务器。最早支持 Servlet 标准的是 JavaSoft 的 Java Web Server，此后，一些其它的基于 Java 的 Web 服务器开始支持标准的 Servlet。

Servlet 使用：

Servlet 对多个请求的接收，主要使用反射，以接收文章相关操作的 servlet 为例：

代码：

```
 * 利用反射获取方法名
 *
 */
request.setCharacterEncoding("utf-8");
response.setContentType("text/html;charset=utf-8");
//1.调用getServletPath()得到方法/insertArticle.do
String servletPath = request.getServletPath();
//去掉/和.do
String methodName=servletPath.substring(1);
methodName=methodName.substring(0, servletPath.length()-4);

try {
    //利用反射调用methodName对应的方法
    Method method=getClass().getDeclaredMethod(methodName, HttpServletRequest.class,HttpServletResponse.class);

    method.invoke(this, request,response);
} catch (Exception e) {
    e.printStackTrace();
    response.sendRedirect("Admin/error.jsp");
}
```

这里就是反射获取需要调用的方法名：

它在 web.xml 上的配置为：

```
<servlet>
  <description></description>
  <display-name>ArticlesServlet</display-name>
  <servlet-name>ArticlesServlet</servlet-name>
  <servlet-class>cn.edu.swu.Servlet.ArticleServlet.ArticlesServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>ArticlesServlet</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

这里就实现了多个请求访问一个 servlet,获取 servletPath(诸如/login.do 等)首先利用 substring 进行字符串的截取，获取出 login,再利用反射调用具体的 Method 对象，然后将 Http 请求和响应作为参数，使用 invoke 方法开始调用 Method 类代表的方法。

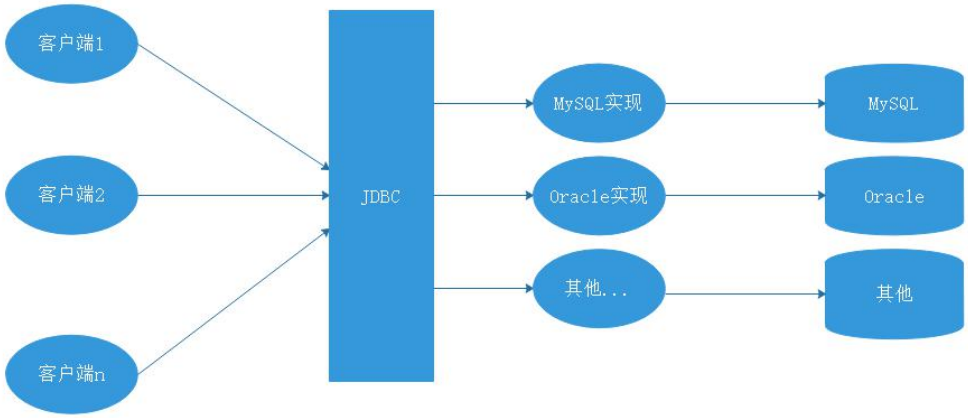
另一个难点基本是依靠文档的使用说明的解决，具体实现的网页：

https://commons.apache.org/proper/commons-fileupload/using.html

4.2.3 JDBC 技术

JDBC 介绍;

JDBC（Java DataBase Connectivity）是 Java 和数据库之间的一个桥梁，是一个规范而不是一个实现，能够执行 SQL 语句。它由一组用 Java 语言编写的类和接口组成。各种不同类型的数据库都有相应的实现，本文中的代码都是采用 MySQL 数据库实现的。



JDBC 为了提高访问效率，减少简介操作等，引入数据库连接池 c3p0.配置数据源：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<c3p0-config>
    <named-config name="MyDB">
        <property name="user">root</property>
        <property name="password">zly970114</property>
        <property name="driverClass">com.mysql.cj.jdbc.Driver</property>
        <property name="jdbcUrl">jdbc:mysql://localhost:3306/mystore?serverTimezone=GMT%2B8</property>
        <property name="acquireIncrement">5</property>
        <property name="initialPoolSize">10</property>
        <property name="minPoolSize">10</property>
        <property name="maxPoolSize">50</property>
        <property name="maxStatements">20</property>
        <property name="maxStatementsPerConnection">5</property>
    </named-config>
</c3p0-config>
```

调用数据源，启动连接池的工具类：

```java
public class JdbcTools {


    private static DataSource dataSource=null;
    static {
        dataSource=new ComboPooledDataSource("MyDB");
    }
    /*
     * 释放
     * 返回数据源的一个connection对象
     */
    public static Connection getConnection() throws Exception {

        return dataSource.getConnection();
    }

    /**
     *
     * 释放资源
     * @param rs
     * @param ps
     * @param con
     */
    public static void release(Connection con) {

        try {
            if(con!=null)
                con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

由于在进行数据库操作时，操作返回都是一个对象集合，一个对象，或者数据库一个相关的值，而在方法中都会进行连接 Connection、predstatement 等相关操作，可以将之提取，写成通用的方法，具体 DAO 实现如下：

首先需要传入一个类型 T，获取类型名代码如下：

```
public class DAO<T> {

    private static  QueryRunner qr = new QueryRunner();
    private Class<T> clazz;

    public DAO() {
        Type superclass = getClass().getGenericSuperclass();
        if (superclass instanceof ParameterizedType) {
            ParameterizedType parameterizedType=(ParameterizedType) superclass;
            Type [] typeArgs=parameterizedType.getActualTypeArguments();
            if(typeArgs!=null && typeArgs.length>0) {
                clazz= (Class<T>) typeArgs[0];
            }
        }
    }
```

获取 T 名以后，则可以使用相应的方法，获取一个 T 对象的结合：

```
    /**
     * 返回T对应的LIST
     */
    public List<T> getForList(String sql,Object...args) {
        Connection con=null;
        try {
            con=JdbcTools.getConnection();
            return qr.query(con, sql, new BeanListHandler<T>(clazz),args);
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            JdbcTools.release(con);
        }
        return null;
    }
```

参数是可变类型参数，可以是具体值，也可以是对象。数据库查询则是使用 dbutils 小工具，进行查询。

获取某一个值：

```
    public <E> E getForValue(String sql,Object...args) {
        Connection con=null;
        try {
            con=JdbcTools.getConnection();
            return (E) qr.query(con, sql, new ScalarHandler(),args);
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            JdbcTools.release(con);
        }
        return null;
```

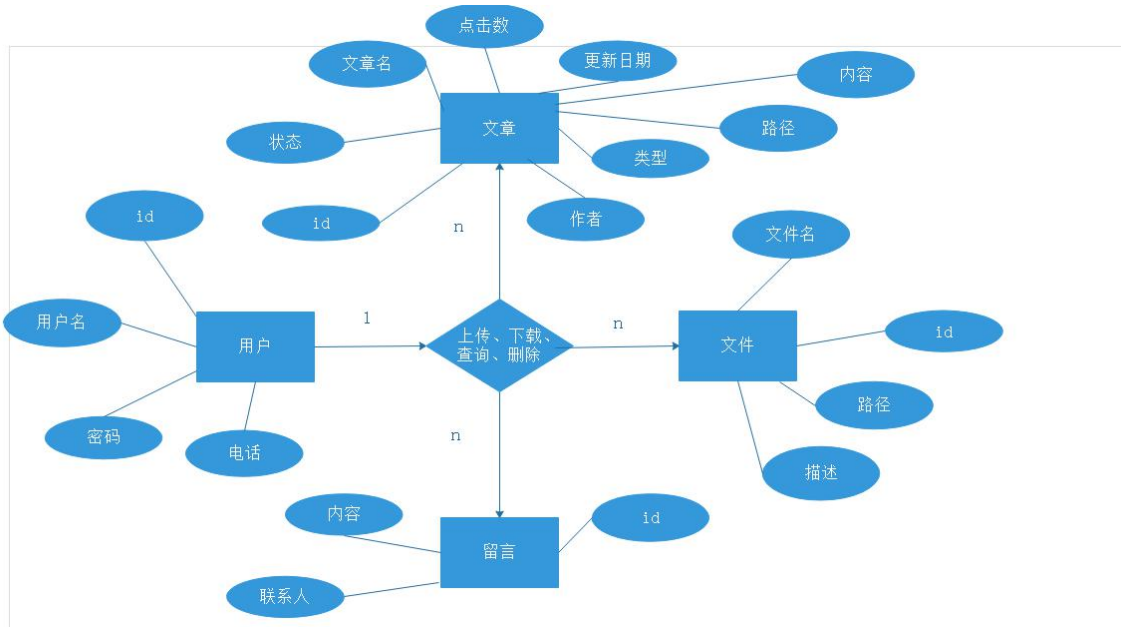
对数据库的修改操作（增、删、改）：

```
/**
 * 封装insert
 * @param sql
 * @param args
 */
public void Update(String sql,Object...args) {
    Connection con=null;
    try {
        con=JdbcTools.getConnection();
        qr.update(con, sql, args);
    } catch (Exception e) {
        e.printStackTrace();
    }finally {
        JdbcTools.release(con);
    }
}
```

4.2.4 MySQL 技术

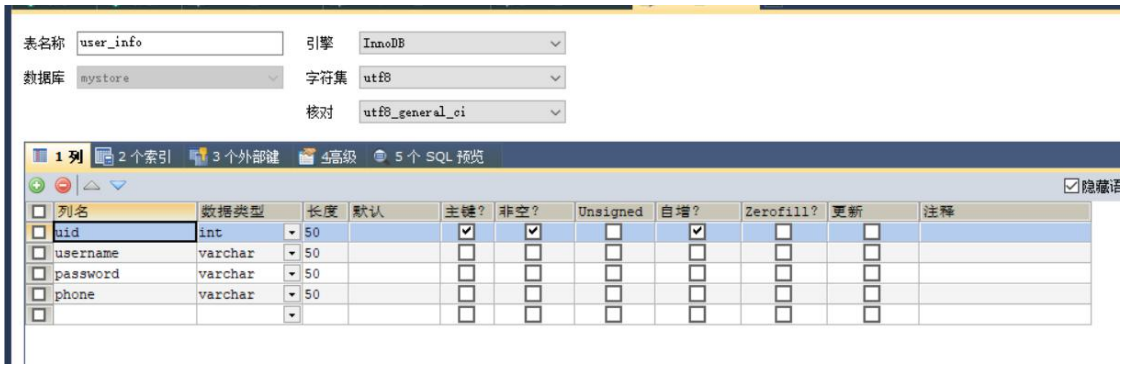
MySQL 作为一种关系型数据库，应用场景广泛，同时由于其免费，非常适合个人网站练手。本项目主要使用 MySQL 作为数据库,其中主要是关于其 E-R 图设计和数据表的建立!

E-R 图：

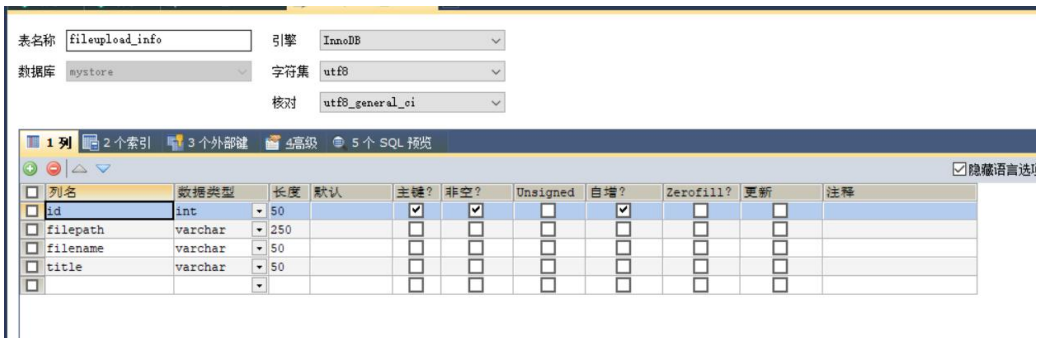


数据表的建立如下：

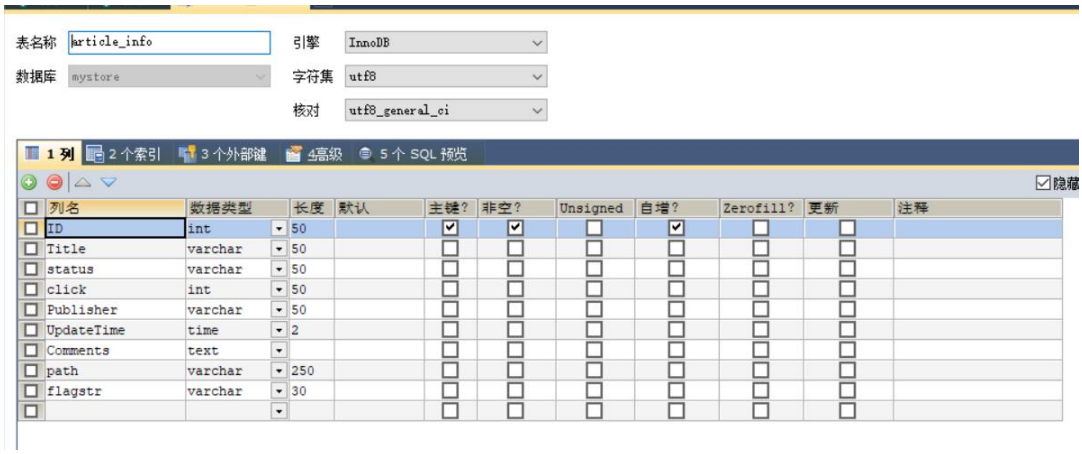
用户表：



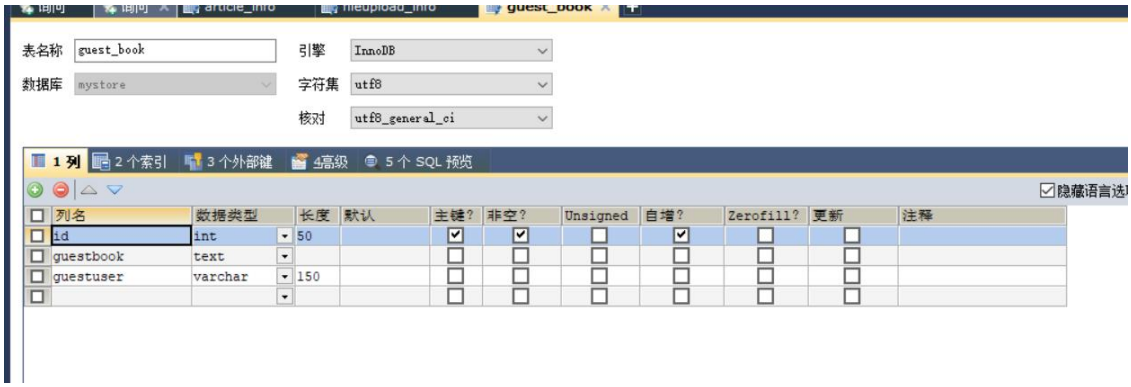
文件信息表：



文章信息表：



留言表：

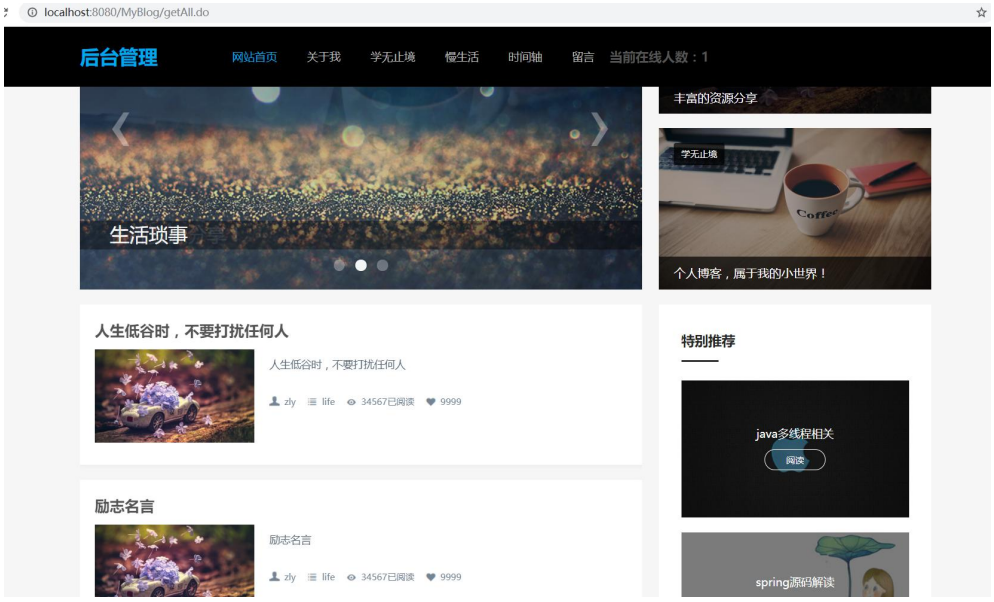


第五章 系统实现

5. 1 系统实现介绍

5.1.前端页面展示：

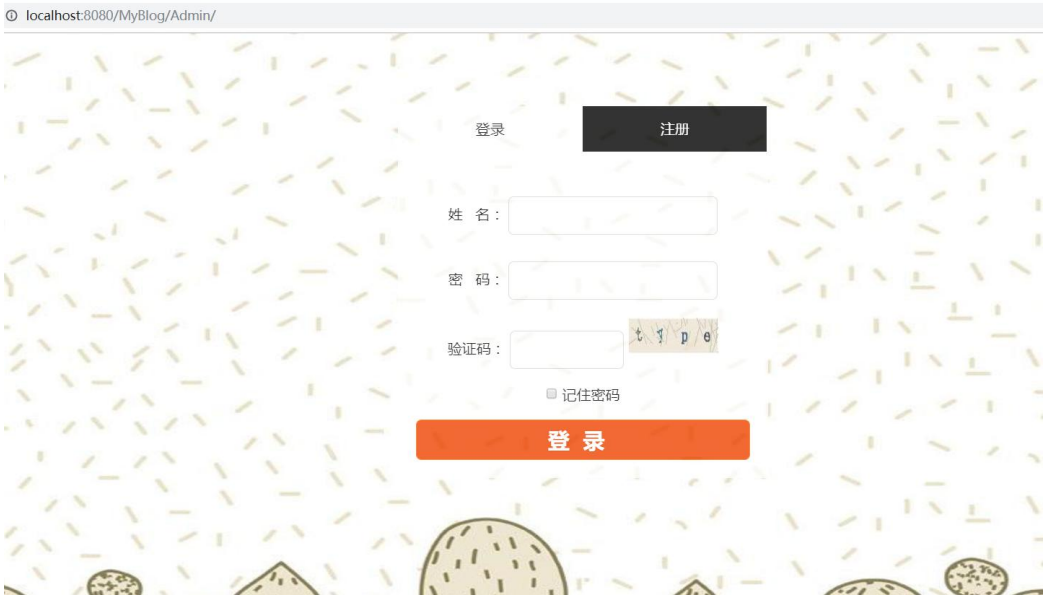
使用 eclipse 运行该项目,其前台个人博客如图：



点击某一个可以进行文章浏览：

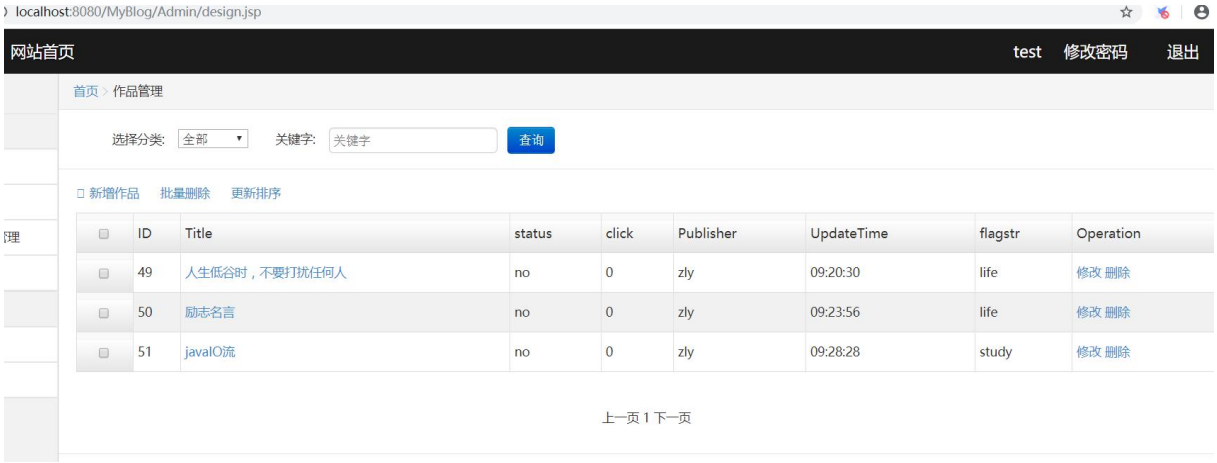
这里使用 servlet+jsp+JavaBean,利用 servlet 接收查询请求，使用 JDBC 查询出的数据库信息分装到 JavaBean 中，发送给前端 jsp 展示。其中在线人数统计是利用了 HttpSessionListener 进行监听创建值。

后台登录注册界面：



同样利用 servlet 获取用户名、密码、验证码等，将其作为参数，利用 JDBC 对数据库进行查询，查看其是否有该用户，存在则跳转后台管理界面；否则不跳转。在注册时需要先检验用户是否已经注册，对已注册的用户需要进行提示！

文章管理界面：



可以看到具体的信息！这里与前台主页的展示基本一样，而删除和修改都会调用相应的 servlet 里面的方法，然后调用 JDBC，实现对数据库的操作 ！

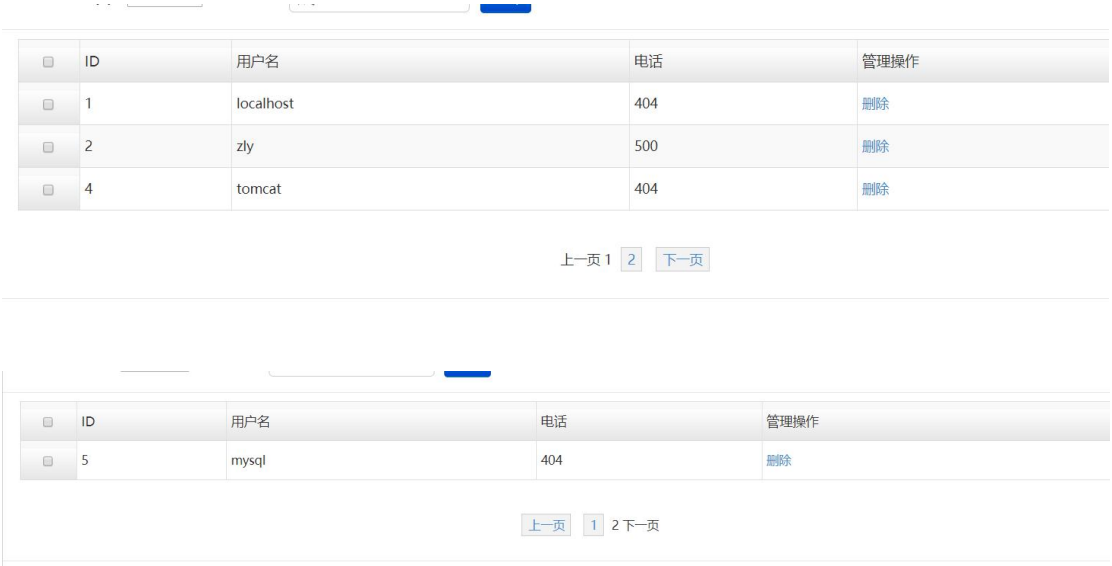
同时这里文章的新增，用到了 java 的文件读写操作

```java
public void MakeHtml(String filePath,String disrPath,String fileName,String content) throws Exception {

    //System.out.println(filePath);
    String tempContext="";
    FileInputStream fileInputStream=new FileInputStream(filePath);//读取模板文件
    int lenght=fileInputStream.available();

    byte[] bytes=new byte[lenght];
    fileInputStream.read(bytes);
    fileInputStream.close();

    tempContext=new String(bytes);
    content=content.replaceAll("\n","<br>" );
    content=content.replaceAll(" "," " );
    //System.out.println(tempContext);
    tempContext=tempContext.replaceAll("##title##", content);
    //System.out.println(tempContext);

    String filepac=fileName+".jsp";
    filepac=disrPath+filepac;
    FileOutputStream fileOutputStream=new FileOutputStream(filepac);
    //System.out.println("文件输出路径: "+filepac);

    byte[] tag_bytes=tempContext.getBytes();
    fileOutputStream.write(tag_bytes);
    fileOutputStream.close();
```

这里先读取模板文件内容，然后再将自己写的内容替换掉里面的标记 title 处，写成一个新的 jsp 文件。

而其中比较难实现的是分页的实现！

分页实现案例，用户管理界面：



这里分页首先要写一个 Page 的 JavaBean,里面包含页数，当前页码，存放一个泛型的 List 集合。每一次查询访问时，都会调用 servlet 中的相应方法，然后用 JDBC 实现信息的存储，最后将信息放给前端展示，这里难点时数据库的访问和在前端展示页面。数据库的访问需要先查询出所有信息的数量，在利用 limit 的 sql 语句进行查询。代码如下：

```java
public PageBeanList<Article> getArticlePage(int pc) {
    int ps=3;//每一页的数量
    long count;
    String sql="SELECT count(*) FROM article_info";
    count=getForValue(sql);
    PageBeanList<Article> pageBeanList=new PageBeanList<Article>();
    pageBeanList.setPs(ps);
    pageBeanList.setPc(pc);
    pageBeanList.setCount(count);
    String sql1="SELECT * FROM article_info LIMIT "+(pc-1)*ps+","+ps;
    List<Article> articles=new ArrayList<Article>();
    articles=getForList(sql1);
    pageBeanList.setBeanList(articles);
    return pageBeanList;
}
```

而前端则需要引入一个通用的 jsp,来用来展示分页，具体代码如下：

```jsp
<c:choose>
    <c:when test="${pbList.pc eq null or pbList.pc eq 1 }">上一页   </c:when>
    <c:otherwise >
        <a href="${pbList.url }pc=${pbList.pc-1}">上一页</a>
    </c:otherwise>
</c:choose>
<c:choose>

    <c:when test="${pbList.tp<6 }">
        <c:set value="1" var="begin"></c:set>
        <c:set value="${pbList.tp }" var="end"></c:set>
    </c:when>
    <c:otherwise>
        <c:set value="${pbList.pc-2 }" var="begin"></c:set>
        <c:set value="${pbList.pc+3 }" var="end"></c:set>

        <c:if test="${pbList.pc<3 }">
            <c:set value="1" var="begin"></c:set>
            <c:set value="6" var="end"></c:set>
        </c:if>
        <c:if test="${pbList.pc+3>pbList.tp }">
            <c:set value="${pbList.tp-5 }" var="begin"></c:set>
            <c:set value="${pbList.tp }" var="end"></c:set>
        </c:if>
    </c:otherwise>
</c:choose>

<c:forEach begin="${begin }" end="${end }" var="page">
    <c:choose>
        <c:when test="${pbList.pc eq page }">${page }</c:when>
        <c:otherwise>
            <a href="${pbList.url }pc=${page}">${page }</a>
        </c:otherwise>
    </c:choose>
</c:forEach>

<c:choose>
    <c:when test="${pbList.pc eq null or pbList.pc eq pbList.tp }">下一页</c:when>
    <c:otherwise>
        <a href="${pbList.url }pc=${pbList.pc + 1}">下一页</a>
    </c:otherwise>
</c:choose>
```

5.2 系统实现的不足

本项目基本实现了个人博客的要求，实现了游客对日志、文章的访问以及对博主的留言，同时博主能够登录后台管理，发表文章，上传文件，管理文章、文件基本信息，留言管理。

同时系统也存在一些不足之处：

1. 新增的文章中间不能加入图片等等，同时在文章中对一些重要的区域无法清晰的表现。
2. 项目运行无法支持多个游客同时访问，当访问数量太多时，会出现访问过慢或者界面错误。
3. 个人文章内没有内嵌评论功能，无法在线看到游客对文章的看法。
4. 对上传的文件没有设置限制类型，没有进行验证。
5. 对留言，注册用户等等没有进行敏感字过滤。
6. 在线人数统计是利用 session 进行估略的统计，没有过滤同一 IP 访问等等。
7. 由于采用的时 servlet 为核心的开发,所以导致开发时配置 web.xml 频繁，开发一个功能，修改部分较多，项目的开发繁琐和部署。