**设计说明书**

# 文档说明

此文档是斯多克科技内部资料，用于内部交流，需要严格保密。

# 2.系统需求分析

## 2.1 需求说明

该系统在性能功能上应达到如下需求：操作简单、界面友好、完全框架式的页面布局，使得新闻的录入工作更简便，许多选项包括新闻分类、新闻列表、新闻出处等只需要点击鼠标就可以完成；另外，跟踪出现的提示信息也让用户随时清楚自己的操作情况。对常见网站的新闻管理的各个方面：新闻录入、浏览、删除、修改、搜索等方面都大体实现，实现了网站对即时新闻的管理要求。

通过相关调查，要求该系统应具有以下功能：

1. 通过网络浏览各类新闻及相关信息。

2. 新闻能分类显示相关信息，一条新闻可以属于多个类。

3. 提供站内新闻的搜索功能。

4. 支持软件下载功能。

5. 为后台管理提供管理入口。

6. 后台编辑个类新闻并且管理新闻信息。

## 2.2 用户功能分析

对于本系统，用户共分为二：会员用户、管理员，各有不同的权限。

（一）会员用户：

此类用户具有以下权限：

1.注册会员。

2.登陆会员。

3.浏览前台网页。

4.阅读已发布新闻。

5.其它特殊权限

（二）管理员：

此类用户具有以下权限：

1. 浏览所有网页。

2. 具有管理普通用户的权限。

3. 管理新闻信息，可供用户浏览，新闻中可包含附件。

5. 管理软件，可以批量上传软件供用户下载。

6. 登陆后可修改个人密码。

7. 可以删除任何一个普通用户的所有信息。

## 2.3 用例图

浏览者和管理员的具体分工操作所涉及到的功能（如图 2.1）所示。



图2.1 用例图

## 2.4 结构图

根据需求分析得出所需要的功能，和结构得出系统结构（如图2.1）所示。



图 2.2 结构图

# 3. 总体设计方案

## 3.1开发工具的选用和介绍

### 3.1.1 JDK环境

JDK环境，这是Java系统编译运行时必须配置的环境，只有在此环境下Java代码才能够执行出结果，并且在本次设计中我们选用的Tomcat服务器的配置也必须在配置好JDK环境之后才能运行，所以本次设计我们选用了1.6的版本进行开发。

### 3.1.2 MySQL

MySQL数据库管理系统是用于关联数据库并将数据保存到所创建的表中，在本次设计中涉及到数据的存储过程，所以必须要有数据库管理系统来进行管理。目前来说MySQL一般比价适用于中小型企业的，如果数据规模非常庞大一般会选择[Oracle](http://baike.baidu.com/view/15020.htm)数据库管理系统进行开发。本次设计数据非常有限所以选择MySQL，除此之外他还有便于学习，成本低，开源等特点。

### 3.1.3 WEB服务器Tomcat

Tomcat是Apache底下的一个项目，它拥有免费、开放源码的特点因此Tomcat成为许多程序开发、调试时所选择的服务器，在本次设计过程中主要使用它来进行程序的调试。同时在本次设计中需要用到JSP和Servlet技术，Tomcat对于这些技术也是兼容的，因此它成为了我对服务器选择的首选。

在本系统中我们选用稳定版本Tomcat6.0进行开发。

### 3.1.4 集成开发工具MyEclipse

是目前最流行的Java集成开发工具。MyEclipse一经出现就以其快速、丰富的特性赢得了认可，其主要的属性有快速的性能、重构性能、快速修复错误、调整组织导入包、弹出窗口进行代码自动装配并且它能够很好地与MySQL、Tomcat进行集成。

本系统中我们选用MyEclipse8.6。

### 3.1.5 Dreamweaver CS5

Adobe Dreamweaver CS5是一款集网页制作和管理网站于一身的所见即所得网页编辑器，Dreamweaver CS5是第一套针对专业[网页设计](http://baike.baidu.com/view/8972.htm)师特别发展的视觉化网页开发工具，利用它可以轻而易举地制作出跨越平台限制和跨越[浏览器](http://baike.baidu.com/view/7718.htm)限制的充满动感的网页。

## 3.2 开发技术的介绍

### 3.2.1 JSP页面技术

JSP产生于Servlet之后，主要是为了解决Servlet中输出HTML代码效率低的问题而产生的。JSP技术比较简单，类似于ASP、PHP这些脚本语言。JSP的技术主要是由以下两个部分组成：

JSP的基本语法：两个注释、3个脚本元素、8个动作指令。

JSP的九大内置对象：request请求对象、response响应对象、pageContext页面上下文对象、session会话对象、application应用程序对象、out输出对象、config配置对象、page页面对象和exception例外对象。

### 3.2.2 Struts2+Spring3+Hibernate技术

Struts通过采用 Java Servlet/JSP 技术，实现了基于Java EEWeb应用的MVC设计模式的应用框架，是MVC经典设计模式中的一个经典产品。Struts 2是Struts的下一代产品，是在 struts 1和WebWork的技术基础上进行了合并的全新的Struts 2框架。Struts 2以WebWork为核心，采用拦截器的机制来处理用户的请求，这样的设计也使得业务逻辑控制器能够与ServletAPI完全脱离开，所以Struts 2可以理解为WebWork的更新产品。

Spring是一个开源框架，它由Rod Johnson创建。它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。简单来说，Spring是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架。

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。

### 3.2.3 HTML标签语言和JavaScript脚本语言的编写

HTML：超级文本标记语言是[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)下的一个应用，也是一种规范，一种[标准](http://baike.baidu.com/view/8079.htm)，它通过标记符号来标记要显示的网页中的各个部分。

Javascript是由Netscape公司创造的一种脚本语言。为便于推广，被定为javascript，但是javascript与java是两门不相干的语言，作用也不一样。作为一门独立的编程语言，javascript可以做很多的事情，但它最主流的应用还是在Web上——创建动态网页。

提要：Javascript是学习脚本语言的首选。她兼容性好，绝大多数浏览器均支持Javascript，而且她功能强大，实现简单方便，入门简单，即使是程序设计新手也可以非常快速容易地使用JavaScript进行简单的编程。

## 3.3 集成开发框架SSH的概述

### 3.3.1 SSH框架简介

集成SSH框架的系统从职责上分为四层：表示层、业务逻辑层、数据持久层和域模块层，以帮助开发人员在短期内搭建结构清晰、可复用性好、维护方便的Web应用程序。其中使用Struts作为系统的整体基础架构，负责MVC的分离，在Struts框架的模型部分，控制业务跳转，利用Hibernate框架对持久层提供支持，Spring做管理，管理Struts和Hibernate。具体做法是：用面向对象的分析方法根据需求提出一些模型，将这些模型实现为基本的Java对象，然后编写基本的DAO(Data Access Objects)接口，并给出Hibernate的DAO实现，采用Hibernate架构实现的DAO类来实现Java类与数据库之间的转换和访问，最后由Spring做管理，管理Struts和Hibernate。

由SSH构建系统的基本业务流程是：在表示层中，首先通过JSP页面实现交互界面，负责接收请求(Request)和传送响应(Response)，然后Struts根据配置文件(struts-config.xml)将ActionServlet接收到的Request委派给相应的Action处理。在业务层中，管理服务组件的Spring IoC容器负责向Action提供业务模型(Model)组件和该组件的协作对象数据处理(DAO)组件完成业务逻辑，并提供事务处理、缓冲池等容器组件以提升系统性能和保证数据的完整性。而在持久层中，则依赖于Hibernate的对象化映射和数据库交互，处理DAO组件请求的数据，并返回处理结果。

采用上述开发模型，不仅实现了视图、控制器与模型的彻底分离，而且还实现了业务逻辑层与持久层的分离。这样无论前端如何变化，模型层只需很少的改动，并且数据库的变化也不会对前端有所影响，大大提高了系统的可复用性。而且由于不同层之间耦合度小，有利于团队成员并行工作，大大提高了开发效率。

### 3.3.2 轻量级J2EE应用框架

轻量级J2EE架构以Spring IoC容器为核心，承上启下。其向上管理来自表现层的Action，向下管理业务逻辑层组件，同时负责管理业务逻辑层所需的DAO对象。各层之间负责传值的是值对象，也就是JavaBean实例。

图2.1精确地描绘了轻量级J2EE架构的大致情形。

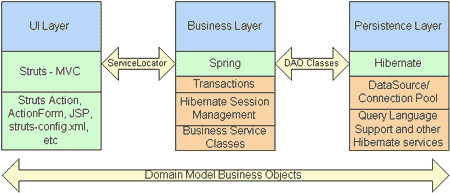


图2.1 轻量级J2EE应用架构

我们可以通过Spring与Struts整合和Spring与Hibernate整合，让Spring管理控制器，同时让Spring管理Hibernate的SessionFactory，实现了整个SSH框架的整合。

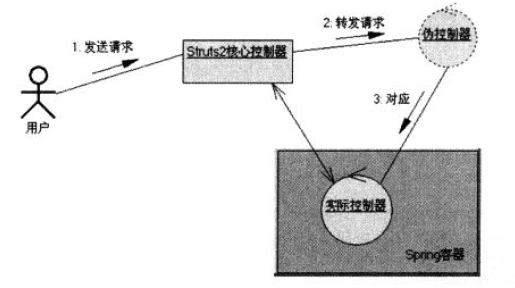


图2.2 Spring管理Action的协作图

# 4.数据库设计

## 4.1设计步骤

1.根据需求分析对所需存储数据进行分类，建立数据库表存储、归类所需数据，确定表中各属性的类型。

2.在数据库表建立完成后还要建立各个表之间的连接关系，确定表与表之间是主外键的关系还是多对多、一对多或是一对一等的关系。

3.分析前台与后台的功能建立用户的权限关系。

## 4.2 数据库表的设计

根据用户需求，共创建了六个表，分别为：管理员信息表admin、会员信息表users、文章信息表newsinfo、评论信息表replys、文章频道表newstypeinfo、会员留言表leavemessage。以下是新闻发布系统的数据库表结构的详细描述：

### 4.2.1 管理员信息表

管理员信息表admin包含着管理员在登录时候所需的账号、密码，还有管理员的信息，管理员的Id作为主键，在查询是可通过Id来查找管理员的信息。（如表4.1）

表4.1 管理员信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段说明 | 类型 | 长度 | 允许空值 | 备注 |
| adminId | 管理员Id | int | 11 | 否 | 主键 |
| adminName | 管理员用户 | varchar | 20 | 否 |  |
| adminPass | 管理员密码 | varchar | 32 | 是 |  |
| adminInfo | 信息 | text |  | 是 |  |

### 4.2.2 会员信息表

会员信息表users包含着会员注册时候所需要的填写的所有信息，以及会员Id设为主键，管理员在查找会员信息时可以根据Id来查找。（如表4.2）

表4.2 会员信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段说明 | 类型 | 长度 | 允许空值 | 备注 |
| usersId | 会员Id | int | 11 | 否 | 主键 |
| usersName | 会员用户 | varchar | 20 | 否 |  |
| usersPass | 会员密码 | varchar | 32 | 是 |  |
| usersEmail | E-mail | varchar | 50 | 是 |  |
| usersInfo | 信息 | text |  | 是 |  |
| realName | 真实姓名 | varchar | 20 | 是 |  |
| sex | 性别 | varchar | 10 | 是 |  |
| phone | 电话号码 | varchar | 20 | 是 |  |
| idNumber | 身份证号 | varchar | 30 | 是 |  |

### 4.2.3 文章信息表

文章信息表newsinfo包含着文章发表时候的全部信息，以及文章Id设为主键用于在查找文章时可以根据此标识来查找出文章的所有信息。同时还在表中设有管理员Id作为外键，用于对管理员信息表进行关联。（如表4.3）

表4.3 文章信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段说明 | 类型 | 长度 | 允许空值 | 备注 |
| newsInfoId | 文章Id | int | 11 | 否 | 主键 |
| newsInfoTitle | 文章标题 | varchar | 100 | 是 |  |
| newsInfoDescribe | 文章描述 | text |  | 是 |  |
| newsInfoContent | 文章内容 | longtext |  | 是 |  |
| newsInfoTime | 创建时间 | datetime |  | 是 |  |
| newsAuthor | 文章作者 | varchar | 40 | 是 |  |
| adminId | 管理员Id | int | 11 | 是 | 外键 |
| newsType | 文章分类 | text |  | 是 |  |
| newsInfoState |  | int | 11 | 是 |  |

### 4.2.4 评论信息表(暂时没用)

评论信息表replys包含着会员评论时候的全部信息，以及评论Id设为主键用于在查找评论时可以根据此标识来查找出评论的所有信息。同时还在表中设有会员Id、文章Id作为外键，用于对会员信息表、文章信息表进行关联。（如表4.4）

表4.4 评论信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段说明 | 类型 | 长度 | 允许空值 | 备注 |
| replytId | 评论Id | int | 11 | 否 | 主键 |
| replytContent | 评论内容 | longtext |  | 是 |  |
| replytTime | 评论时间 | datetime |  | 是 |  |
| usersId | 会员Id | int | 11 | 是 | 外键 |
| newsInfoId | 文章Id | int | 11 | 是 | 外键 |

### 4.2.5 文章频道表

文章频道表newstype主要是用于对文章的分类，管理员在发布新闻时需要对新闻进行分类存放到不同的频道，以便会员可以通过在查看文章频道来查找相关的文章。（如表4.5）

表4.5 文章频道表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段说明 | 类型 | 长度 | 允许空值 | 备注 |
| newsTypeId | 频道Id | int | 11 | 否 | 主键 |
| newsTypeName | 频道名 | varchar | 200 | 是 |  |
| newsTypeDescripe | 频道描述 | text |  | 是 |  |

### 4.2.6 会员留言表（暂时没用）

会员留言表leavemessage主要是包含会员留言内容，以及留言Id设为主键以便管理员可以通过留言Id来查找留言，有需要时候可进行删除管理，此表还包含有会员Id设为外键用于标识是哪位会员进行的留言。（如表4.6）

表4.6 会员留言表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段说明 | 类型 | 长度 | 允许空值 | 备注 |
| leaveMessageId | 留言Id | int | 11 | 否 | 主键 |
| leaveMessageContent | 留言内容 | text |  | 是 |  |
| leaveMessageTime | 留言时间 | datetime |  | 是 |  |
| userId | 会员Id | int | 11 | 是 | 外键 |

### 4.2.6 新闻附件表

新闻附件表news\_attachment主要用于保存与新闻相关的附件，新闻ID作为外键，与文章信息表关联。当新闻ID为0时，表示保存的是软件信息，其中attachment\_content域为空，实际软件被保存在工程的softwares目录中。

表4.7 新闻附件表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段说明 | 类型 | 长度 | 允许空值 | 备注 |
| attachment\_id | 附件Id | int | 10 | 否 | 主键 |
| news\_id | 新闻Id | text |  | 是 | 外键 |
| attachment\_name | 附件名称 | varchar | 50 | 是 |  |
| attachment\_content | 附件内容 | longblob |  | 是 |  |

### 4.2.7 会员权限表

会员权限表permission将会员分成几个不同的等级，每个等级的会员在个人网站的操作上有不同的权限，例如普通会员只能上传新闻图片、钻石会员增加了上传表格的权限而白金会员还有增加频道类型的权限。每个权限由身份证ID和授权ID共同决定。

表4.8 会员权限表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段说明 | 类型 | 长度 | 允许空值 | 备注 |
| permissionId | 权限Id | int | 11 | 否 | 主键 |
| idNumber | 身份证Id | varchar | 30 | 否 | 外键 |
| authenticationId | 授权Id | varchar | 50 | 否 |  |
| permissionLevel | 权限等级 | int | 10 | 否 |  |
| cmt | 备注 | text |  | 是 |  |

## 4.3 数据关系ER图

ER图主要是用于显示出每个对象的属性以及每个对象之间的联系关系，对于从数据表结构到代码的实现有一定的指导作用。本次系统开发的ER图结构。（如图4.1）

 图4.1 ER图

# 代码设计

## 5.1 系统开发准备工作

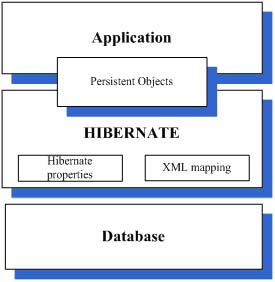
### 5.1.1 JSP文件的导入

在系统开发前我们需要先倒入本项目所需要的JSP模板，这里需要说明的是在一个实际的项目开发过程中会分为负责JSP界面设计的美编，以及负责JAVA软件开发的软件工程师。在本次设计中我们主要是完成软件设计部分的功能，所以开发前需导入一个JSP文件模板。

### 5.1.2 Hibernate数据库连接配置

Hibernate可以使用映射文件和映射类实现ORM（对象/关系数据库映射），将每个class元素映射一个PO（持久化类），从而维护持久化类和数据库表之间的对应关系，也可以理解成持久化类属性与数据表列之间的对应关系。

Hibernate推荐使用数据源来管理数据库请求，支持第三方的连接池，官方推荐的连接池是C3P0,Proxool,以及DBCP。数据源会负责维持一个数据连接池，当程序创建数据源实例时，系统会一次性创建多个数据库连接，并保存在连接池中。当程序需要访问数据库时，只需从连接池中取出一个空闲的数据库连接；当访问结束后，只需将连接归还给连接池。从而避免频繁获取连接、关闭连接，保证最好的性能。



如图 5.1 Hibernate体系结构

由上图可知Hibernate的作用是为数据库的连接提供可扩展的接口，当系统选择不同类型的数据库时，只需采用不同的驱动便可连接上不同的数据库。为系统提高更多的灵活性。

本系统采用MySQL的JDBC驱动进行数据库连接，采用Hibernate推荐的C3P0数据源。整个Hibernate配置文件如下：

<?xml version=*'1.0'* encoding=*'UTF-8'*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<!-- Generated by MyEclipse Hibernate Tools. -->

<hibernate-configuration>

<session-factory>

<property name=*"dialect"*>

org.hibernate.dialect.MySQLInnoDBDialect

</property>

<property name=*"connection.url"*> jdbc:mysql://localhost:3306/news\_all?useUnicode=true&amp;characterEncoding=UTF-8

</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.driver\_class"*>

com.mysql.jdbc.Driver

</property>

<property name=*"myeclipse.connection.profile"*>MySQLdriver</property>

<property name=*"connection.password"*>qxw</property>

<!-- 指定连接池里最大连接数 -->

<property name=*"hibernate.c3p0.max\_size"*>20</property>

<!-- 指定连接池里最小连接数 -->

<property name=*"hibernate.c3p0.min\_size"*>1</property>

<!-- 指定连接池里连接的超时时长 -->

<property name=*"hibernate.c3p0.timeout"*>5000</property>

<!-- 指定连接池里最大缓存多少个Statement对象 -->

<property name=*"hibernate.c3p0.max\_statements"*>100</property>

<property name=*"hibernate.c3p0.idle\_test\_period"*>3000</property>

<property name=*"hibernate.c3p0.acquire\_increment"*>2</property>

<property name=*"hibernate.c3p0.validate"*>true</property>

<property name=*"hibernate.show\_sql"*>true</property>

<property name=*"hibernate.format\_sql"*>true</property>

<mapping class=*"org.news.model.Admin"* />

<mapping class=*"org.news.model.NewsAttachment"* />

<mapping class=*"org.news.model.NewsInfo"* />

<mapping class=*"org.news.model.NewsType"* />

<mapping class=*"org.news.model.LeaveMessage"* />

<mapping class=*"org.news.model.Users"* />

</session-factory>

</hibernate-configuration>

## 5.2 SSH框架实现过程

在本次设计中采用的是SSH框架作为系统的搭建框架，采用这种模式的好处是可以使每一层次的作用分工明确，构建灵活、易于扩展。它强制性的使[应用程序](http://baike.baidu.com/view/330120.htm)的[输入](http://baike.baidu.com/view/700406.htm)、[处理](http://baike.baidu.com/view/989420.htm)和[输出](http://baike.baidu.com/view/700402.htm)分开。使用MVC应用程序被分成三个核心部件：[模型](http://baike.baidu.com/view/96500.htm)（model）、[视图](http://baike.baidu.com/view/71981.htm)（model）、[控制器](http://baike.baidu.com/view/122229.htm)（controller）。本文采用的是JSP+Action+Service+DAO四层模式来搭建整个框架。

JSP为view层，作为视图，用于在浏览器显示页面。Action为controller层，作为控制器，用于实现前台后台的业务数据交互以及逻辑设计。Service实现业务逻辑，为控制器提供各种服务。DAO为model层，作为模型用于从数据库获取数据，这里要指出DAO模式与传统的model层不同，它不仅承载数据，而且包含了内在的逻辑和操作。

本次设计的总体实现过程（如图：5.2）。

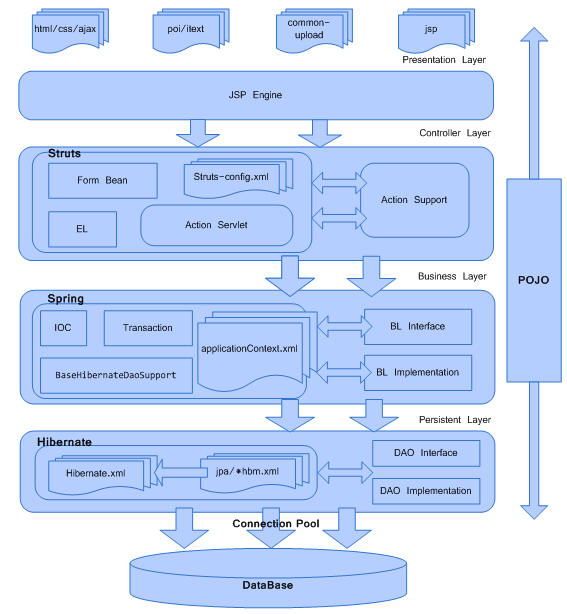


图5.2 SSH框架实现过程

## 5.3系统功能具体实现思想

在系统运行时会涉及到一个通过浏览器对网页的操作，而后产生一系列的请求通过底层代码去调用所需的数据去完成对每个请求的响应的过程。在本系统中这一个过程凡是涉及到与数据库交互数据时都会按照MVC模式流程来完成。这也就使每一层都必须清楚自己所负责的功能。

在本系统的设计过程中我以先后顺序为：model，DAO，Service，Action，JSP的分层顺序搭建起MVC模式。然后在数据库中完成增、删、改查等功能。下面我以某些关键功能的例子来说明系统的实现思想。

### 5.3.1 对信息的批量删除

我们以文章的批量删除为例，其他的删除功能，以及单条记录的删除可以仿照此例实现。

（1）model的建立，我们首先把文章的所有所有属性封装在一个类里面，类名为newsInfo。这个类的属性是与数据库中的表的内容相匹配的，其中类中的Id可以做为该类的对象的索引，此外model中除了拥有属性外还包含了get、set方法用于获取和设置该类对象的属性。

（2）DAO层，该层主要是负责逻辑操作以及与数据库的数据交互，所以首先要创建数据库连接对象，与数据库连接。因为要实现删除功能，所以在DAO层中就定义了一个删除的方法，以批量删除为例，因为为批量删除，所以需要用数组对传来需删除的Id数据进行接收，然后返回boolean类型进行判断，实现代码如下：

/\*\*

\* 批量删除

\* **@param** newsInfoIds

\* **@return** 是否成功

\*/

**public** **boolean** deleteNewsInfo(**int**[] newsInfoIds) {

**boolean** result = **true**;

**for** (**int** i = 0; i < newsInfoIds.length; i++) {

**try** {

getHibernateTemplate().delete(searchNewsInfo(newsInfoIds[i]));

*log*.debug("delete successful");

} **catch** (RuntimeException re) {

*log*.error("delete failed", re);

result = **false**;

**throw** re;

}

}

**return** result;

}

1. Service层通过Spring的Ioc获取DAO，实现批量删除的事务逻辑，将结果返回，实现代码如下：

/\*\*

\* 批量删除服务

\* **@param** newsInfoIds

\* **@return**

\*/

**public** **boolean** deleteNewsInfo(**int**[] newsInfoIds) {

**for** (**int** i=0;i<newsInfoIds.length;++i){

//有附件才删除附件

**if**(attachmentDAO.findNewsAttachmentByNewsId(newsInfoIds[i]).size()>0&&!attachmentDAO.deleteNewsAttachmentByNewsId(newsInfoIds[i])){

**return** **false**;

}

}

**return** newsInfoDAO.deleteNewsInfo(newsInfoIds);

}

1. Action层，本层的作用在于从前台网页获取参数再调用Service层中所提供的方法进行逻辑操作，然后操作结果返回前台网页。需要说明的是在Action中使用getter和setter方法与JSP页面进行数据交换，通过不同的返回值跳转到不同的页面。在本例中批量删除的实现代码如下：

/\*\*

\* 删除新闻信息

\* **@return**

\*/

**public** String delete(){

**int** currentPage = 1 ; // 为当前所在的页，默认在第1页

**int** lineSize = 20; // 每次显示的记录数

String page = pg;

**try**{

currentPage = Integer.*parseInt*(cp) ;

} **catch**(Exception e) {}

**try**{

lineSize = Integer.*parseInt*(ls) ;

} **catch**(Exception e) {}

**int**[] newsId = **new** **int**[1];

newsId[0] = pid;

**try** {

**if** (service.deleteNewsInfo(newsId)) {

setMsg(MessageUtil

.*get*("newsinfo.delete.true"));

} **else** {

setMsg(MessageUtil

.*get*("newsinfo.delete.false"));

}

//将参数转化成属性传给下个页面

setPg(page);

setCp(""+currentPage);

setLs(""+lineSize);

**return** *SUCCESS*;

} **catch** (Exception e) {

e.printStackTrace();

}

**return** *ERROR*;

}

1. JSP层，在这层中与Servlet层有着相同的域来存放参数和内容彼此进行交互。JSP层的作用主要是在HTML网页页面设计的基础上添加了可以实现编写JAVA代码的功能，同时也能插入javascript脚本，通过该脚本可以编写方法各种功能，在事件促发后可调用该方法完成操作。实现代码如下：

**function** **doDelete**(){

**if**(window.confirm("确认删除？")){

**return** **true** ;

} **else** {

**return** **false** ;

}

}

<a href="NewsInfo\_delete?pid=${newsvo.newsInfoId}&cp=${page}&ls=${size}&pg=${url}" onclick="return doDelete()">删除</a></span></td>

如此一来，一个批量删除的功能就能通过前台网页JSP页面获取删除操作经过Action层、Service层、DAO层对数据库中的文章进行删除。

### 5.3.2 使用Spring进行依赖注入和声明式事务控制

Spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

◆轻量——从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的：典型地，Spring应用中的对象不依赖于Spring的特定类。

◆控制反转——Spring通过一种称作控制反转（IoC）的技术促进了松耦合。当应用了IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。你可以认为IoC与JNDI相反——不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

◆面向切面——Spring提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（例如审计（auditing）和事务（transaction）管理）进行内聚性的开发。应用对象只实现它们应该做的——完成业务逻辑——仅此而已。它们并不负责（甚至是意识）其它的系统级关注点，例如日志或事务支持。

◆容器——Spring包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，你可以配置你的每个bean如何被创建——基于一个可配置原型（prototype），你的bean可以创建一个单独的实例或者每次需要时都生成一个新的实例——以及它们是如何相互关联的。然而，Spring不应该被混同于传统的重量级的EJB容器，它们经常是庞大与笨重的，难以使用。

◆框架——Spring可以将简单的组件配置、组合成为复杂的应用。在Spring中，应用对象被声明式地组合，典型地是在一个XML文件里。Spring也提供了很多基础功能（事务管理、持久化框架集成等等），将应用逻辑的开发留给了你。

在本系统中，我们主要使用Spring将DAO、Service注入到ServiceImpl和Action中，同时使用AOP完成了事务的管理，配置文件如下：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop-3.0.xsd"*>

<!-- 配置sessionFactory -->

<bean id=*"sessionFactory"*

class=*"org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"*>

<property name=*"configLocation"* value=*"classpath:hibernate.cfg.xml"* />

</bean>

<!-- 配置事务管理器 -->

<bean id=*"transactionManager"*

class=*"org.springframework.orm.hibernate3.HibernateTransactionManager"*>

<property name=*"sessionFactory"*>

<ref bean=*"sessionFactory"* />

</property>

</bean>

<!-- 配置事务的传播特性 -->

<tx:advice id=*"txAdvice"* transaction-manager=*"transactionManager"*>

<tx:attributes>

<tx:method name=*"add\*"* propagation=*"REQUIRED"* />

<tx:method name=*"delete\*"* propagation=*"REQUIRED"* />

<tx:method name=*"update\*"* propagation=*"REQUIRED"* />

<tx:method name=*"\*"* read-only=*"true"* timeout=*"60"*/>

</tx:attributes>

</tx:advice>

<tx:annotation-driven transaction-manager=*"transactionManager"* proxy-target-class=*"true"*/>

<!-- 那些类的哪些方法参与事务 -->

<aop:config>

<aop:pointcut id=*"allManagerMethod"*

expression=*"execution(\* org.news.service.impl.\*.\*(..))"* />

<aop:advisor pointcut-ref=*"allManagerMethod"* advice-ref=*"txAdvice"* />

</aop:config>

<!-- 定义DAO Bean-->

<bean id=*"adminDAO"* class=*"org.news.dao.AdminHibernateDAO"*>

<!-- 注入持久化操作所需的SessionFactory -->

<property name=*"sessionFactory"* ref=*"sessionFactory"*/>

</bean>

<bean id=*"attachmentDAO"* class=*"org.news.dao.NewsAttachmentHibernateDAO"*>

<!-- 注入持久化操作所需的SessionFactory -->

<property name=*"sessionFactory"* ref=*"sessionFactory"*/>

</bean>

<bean id=*"newsInfoDAO"* class=*"org.news.dao.NewsInfoHibernateDAO"*>

<!-- 注入持久化操作所需的SessionFactory -->

<property name=*"sessionFactory"* ref=*"sessionFactory"*/>

</bean>

<bean id=*"newsTypeDAO"* class=*"org.news.dao.NewsTypeHibernateDAO"*>

<!-- 注入持久化操作所需的SessionFactory -->

<property name=*"sessionFactory"* ref=*"sessionFactory"*/>

</bean>

<bean id=*"usersDAO"* class=*"org.news.dao.UsersHibernateDAO"*>

<!-- 注入持久化操作所需的SessionFactory -->

<property name=*"sessionFactory"* ref=*"sessionFactory"*/>

</bean>

<bean id=*"adminService"* class=*"org.news.service.impl.AdminServiceImpl"*>

<property name=*"adminDAO"* ref=*"adminDAO"*></property></bean>

<bean id=*"attachmentService"* class=*"org.news.service.impl.AttachmentServiceImpl"*>

<property name=*"attachmentDAO"* ref=*"attachmentDAO"*></property></bean>

<bean id=*"newsInfoService"* class=*"org.news.service.impl.NewsInfoServiceImpl"*>

<property name=*"newsInfoDAO"* ref=*"newsInfoDAO"*></property>

<property name=*"adminDAO"* ref=*"adminDAO"*></property>

<property name=*"attachmentDAO"* ref=*"attachmentDAO"*></property></bean>

<bean id=*"newsTypeService"* class=*"org.news.service.impl.NewsTypeServiceImpl"*>

<property name=*"newsTypeDAO"* ref=*"newsTypeDAO"*></property>

<property name=*"infoService"* ref=*"newsInfoService"*></property></bean>

<bean id=*"userService"* class=*"org.news.service.impl.UserServiceImpl"*>

<property name=*"userDAO"* ref=*"usersDAO"*></property></bean>

<bean id=*"adminAction"* class=*"org.news.action.AdminAction"*>

<property name=*"adminService"* ref=*"adminService"*></property></bean>

<bean id=*"adminLogin"* class=*"org.news.action.AdminLogin"*>

<property name=*"adminService"* ref=*"adminService"*></property></bean>

<bean id=*"downloadAttachmentAction"* class=*"org.news.action.DownloadAttachmentAction"*>

<property name=*"service"* ref=*"attachmentService"*></property></bean>

<bean id=*"newsInfoAction"* class=*"org.news.action.NewsInfoAction"*>

<property name=*"service"* ref=*"newsInfoService"*></property>

<property name=*"typeService"* ref=*"newsTypeService"*></property>

<property name=*"attiService"* ref=*"attachmentService"*></property></bean>

<bean id=*"newsTypeAction"* class=*"org.news.action.NewsTypeAction"*>

<property name=*"newstypeService"* ref=*"newsTypeService"*></property></bean>

<bean id=*"softwareAction"* class=*"org.news.action.SoftwareAction"*>

<property name=*"service"* ref=*"attachmentService"*></property></bean>

<bean id=*"userLoginAction"* class=*"org.news.action.UserLoginAction"*>

<property name=*"userService"* ref=*"userService"*></property></bean>

<bean id=*"userRegisterAction"* class=*"org.news.action.UserRegisterAction"*>

<property name=*"userService"* ref=*"userService"*></property></bean>

<bean id=*"userAction"* class=*"org.news.action.UsersAction"*>

<property name=*"userService"* ref=*"userService"*></property></bean>

<bean id=*"validateNameAction"* class=*"org.news.action.ValidateNameAction"*>

<property name=*"userService"* ref=*"userService"*></property>

<property name=*"adminService"* ref=*"adminService"*></property>

<property name=*"newstypeService"* ref=*"newsTypeService"*></property></bean>

<bean id=*"newsInterfaceAction"* class=*"org.news.action.NewsInterfaceAction"*>

<property name=*"service"* ref=*"newsInfoService"*></property>

<property name=*"typeService"* ref=*"newsTypeService"*></property>

<property name=*"attiService"* ref=*"attachmentService"*></property>

</bean>

</beans>

### 使用Struts实现MVC模式

Struts是Apache 基金会Jakarta 项目组的一个Open Source 项目，它采用MVC模式，能够很好地帮助java 开发者利用J2EE开发Web应用。和其他的java架构一样，Struts 也是面向对象设计，将MVC模式"分离显示逻辑和业务逻辑"的能力发挥得淋漓尽致。Struts 2是Struts的下一代产品，以WebWork为核心，采用拦截器的机制来处理用户的请求，这样的设计也使得业务逻辑控制器能够与ServletAPI完全脱离开，所以Struts 2可以理解为WebWork的更新产品。

当Web容器收到请求（HttpServletRequest）它将请求传递给一个标准的的过滤链包括流程（ActionContextCleanUp）过滤器，然后经过Other filters(SiteMesh ,etc),

接下来需要调用FilterDispatcher核心控制器，然后它调用ActionMapper确定请求那个Action，ActionMapper返回一个收集Action详细信息的ActionMaping对象。

接下来FilterDispatcher将控制权委派给ActionProxy,ActionProxy调用配置管理器(ConfigurationManager) 从配置文件中读取配置信息(struts.xml)，然后创建ActionInvocation对象，ActionInvocation在调用Action之前会依次的调用所用配置拦截器（Interceptor N） 一旦执行结果返回结果字符串ActionInvocation负责查找结果字符串对应的(Result）然后执行这个Result Result会调用一些模版（JSP）来呈现页面，之后拦截器(Interceptor N)会在被执行(顺序和Action执行之前相反)最后响应(HttpServletResponse)被返回在web.xml中配置的那些过滤器和（核心控制器）（FilterDispatcher）。

本系统使用Struts实现了整个请求/响应流的MVC分配过程，同时还实现了输入校检、权限检查、文件上传的类型和大小过滤、国标化资源文件、AJAX进行重复性校验和返回值JSON序列化等功能。配置文件如下：

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">

<struts>

<package name=*"default"* namespace=*"/"* extends=*"struts-default"*>

<!-- 通用的Action，\*表示可以处理任意的请求 -->

<action name=*"\*"*>

<!-- 此处{1}表示前面第一个\*代表的字符串 -->

<result>/WEB-INF/pages/{1}.jsp</result>

</action>

<action name=*"userRegister"*

class=*"userRegisterAction"*>

<result name=*"success"*>

<param name=*"location"*>/WEB-INF/pages/regist\_do.jsp</param>

<param name=*"refresh"*>2;URL=/WEB-INF/pages/login.jsp</param>

</result>

<result name=*"error"*>

/WEB-INF/pages/regist\_do.jsp

</result>

<result name=*"input"*>/WEB-INF/pages/regist.jsp</result>

</action>

<action name=*"userLogin"*

class=*"userLoginAction"*>

<result name=*"success"*>/WEB-INF/pages/welcome.jsp</result>

<result name=*"login"*>/WEB-INF/pages/login.jsp</result>

</action>

<action name=*"uploadAction"*

class=*"org.news.action.UploadAction"* method=*"upload"*>

</action>

<action name=*"downloadAttachmentAction"* method=*"download"*

class=*"downloadAttachmentAction"*>

</action>

</package>

<!-- 全局国际化 -->

<constant name=*"struts.multipart.maxSize"* value=*"1073741824"*/>

<constant name=*"struts.custom.i18n.resources"* value=*"mess"*></constant>

<constant name=*"struts.i18n.encoding"* value=*"UTF-8"*></constant>

<constant name=*"struts.devMode"* value=*"true"*></constant>

<constant name=*"struts.multipart.saveDir"* value=*"/tmp"*></constant>

<include file=*"struts-manage.xml"* />

<!-- 接口包 -->

<package name=*"interface"* namespace=*"/interface"*

extends=*"json-default"*>

<action name=*"acquireMenu"*

class=*"newsInterfaceAction"* method=*"acquireMenu"*>

<result type=*"json"*>

<param name=*"noCache"*>true</param>

<param name=*"contentType"*>text/html</param>

</result>

</action>

<action name=*"acquireNewsList"*

class=*"newsInterfaceAction"* method=*"acquireNewsList"*>

<result type=*"json"*>

<param name=*"noCache"*>true</param>

<param name=*"contentType"*>text/html</param>

</result>

<result type=*"plainText"* name=*"error"*>

<param name=*"location"*>/WEB-INF/front/error.ftl</param>

<param name=*"charSet"*>UTF-8</param>

</result>

</action>

<action name=*"acquireNewsInfo"*

class=*"newsInterfaceAction"* method=*"acquireNewsInfo"*>

<result type=*"json"*>

<param name=*"noCache"*>true</param>

<param name=*"contentType"*>text/html</param>

</result>

<result type=*"plainText"* name=*"error"*>

<param name=*"location"*>/WEB-INF/front/error.ftl</param>

<param name=*"charSet"*>UTF-8</param>

</result>

</action>

<action name=*"acquireSoftwarelist"*

class=*"newsInterfaceAction"* method=*"acquireSoftwarelist"*>

<result type=*"json"*>

<param name=*"noCache"*>true</param>

<param name=*"contentType"*>text/html</param>

</result>

</action>

</package>

</struts>

# 6. 系统调试

## 6.1 前台页面调试

新闻发布系统有两种用户：会员用户（包含游客）和管理员，分别有不同的权限，但都是从同一界面登陆，此系统首页采用DIV+CSS框架，使得页面看起来美观舒适。能自动获取系统当前时间，展示新闻列表内容和软件等信息。

用户只要在浏览器地址栏输入改项目的地址，即可访问该系统的相关信息；需要说明的是：因为该系统并未真正发布成正规的网站，只是将项目部署到了Tomcat服务器上面；其他机子需要访问必须在同一个局域网中访问，在浏览器地址栏中输入：http://服务器机子地址：8080/Stock/ 既可访问项目。其中登录URL为8080/ Stock /login，还可以进行注册。

登录时进行验证，用户名以字母开头，长度在6到18之间；密码长度在6到18之间。验证失败弹出提示框。



图6.1 前台会员登录



图6.2 登录验证



图6.3登录成功



图6.4 登录失败

在登陆界面下方有个链接可以进行注册，注册界面如下图所示。在填写时，系统会自动对每一项进行验证。

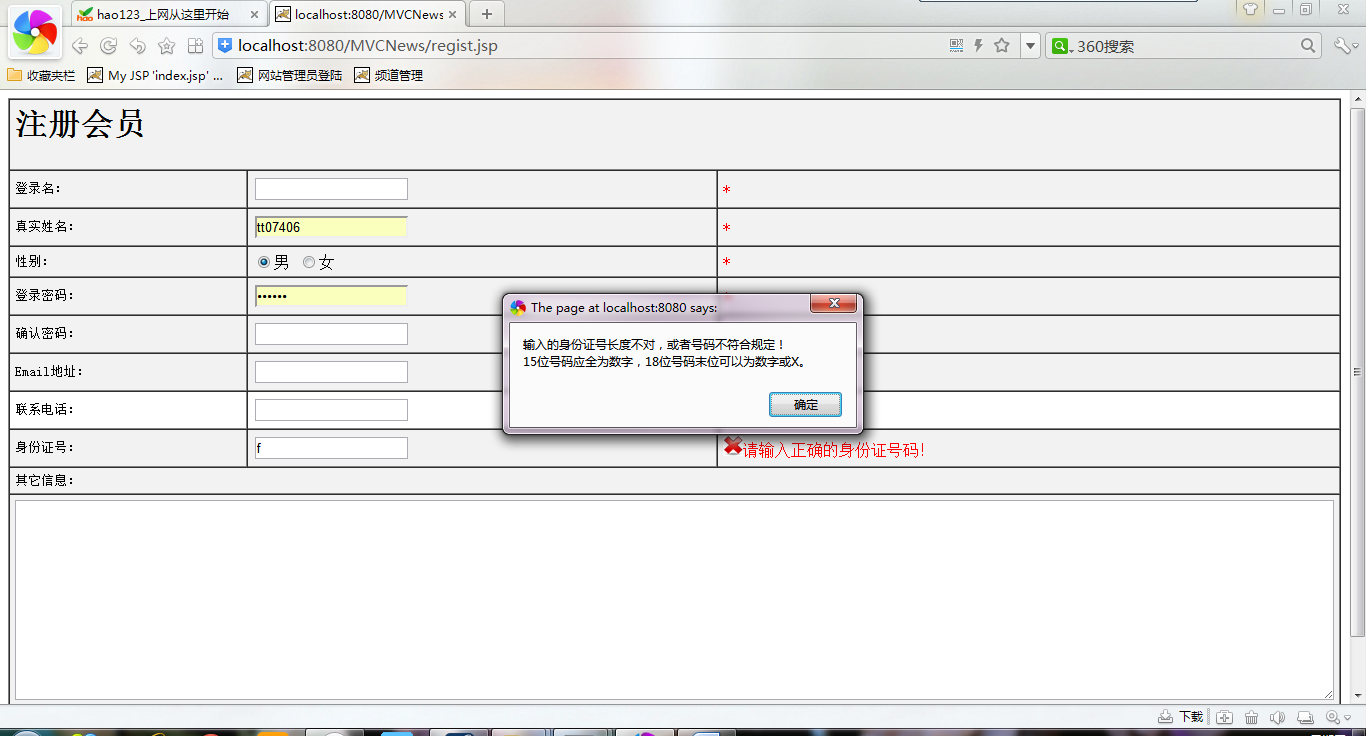


图6.5 会员注册页面

前台输入的页面会进行Xss和SQL过滤，实现网站的整体安全。

## 6.2 后台管理员页面调试



图6.6 后台登录

后台登录界面地址为8080/Stock/manage/login。在该界面输入用户名密码后，完成登录验证后就能进入管理员操作界面（如图6.7）。



图6.7 管理员操作界面

在管理员界面中点击左边栏添加新闻操作可以跳转到添加新闻页面（如图6.8）可以添加新闻，成功后可以在将新闻发布到前台页面，可供用户访问。可以上传多个附件，只需点击右边的“添加”按钮。还可以选择本地图片或者网络图片上传到正文里。



图6.8 添加新闻

点击新闻列表便可以查看所有的新闻，还可以进行新闻的查找和翻页操作。点击标题链接可以查看新闻详情，同样的，修改和删除链接可以实现相应的操作。

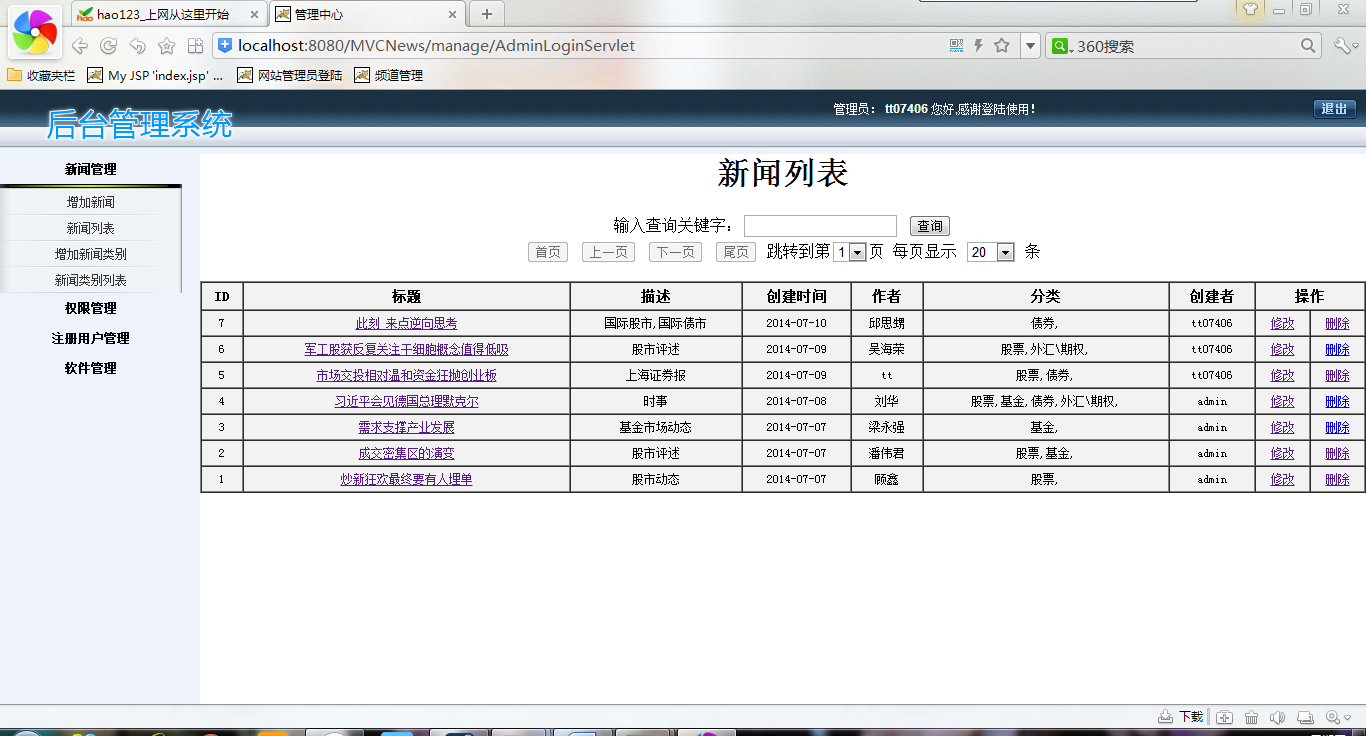


图6.9 新闻列表

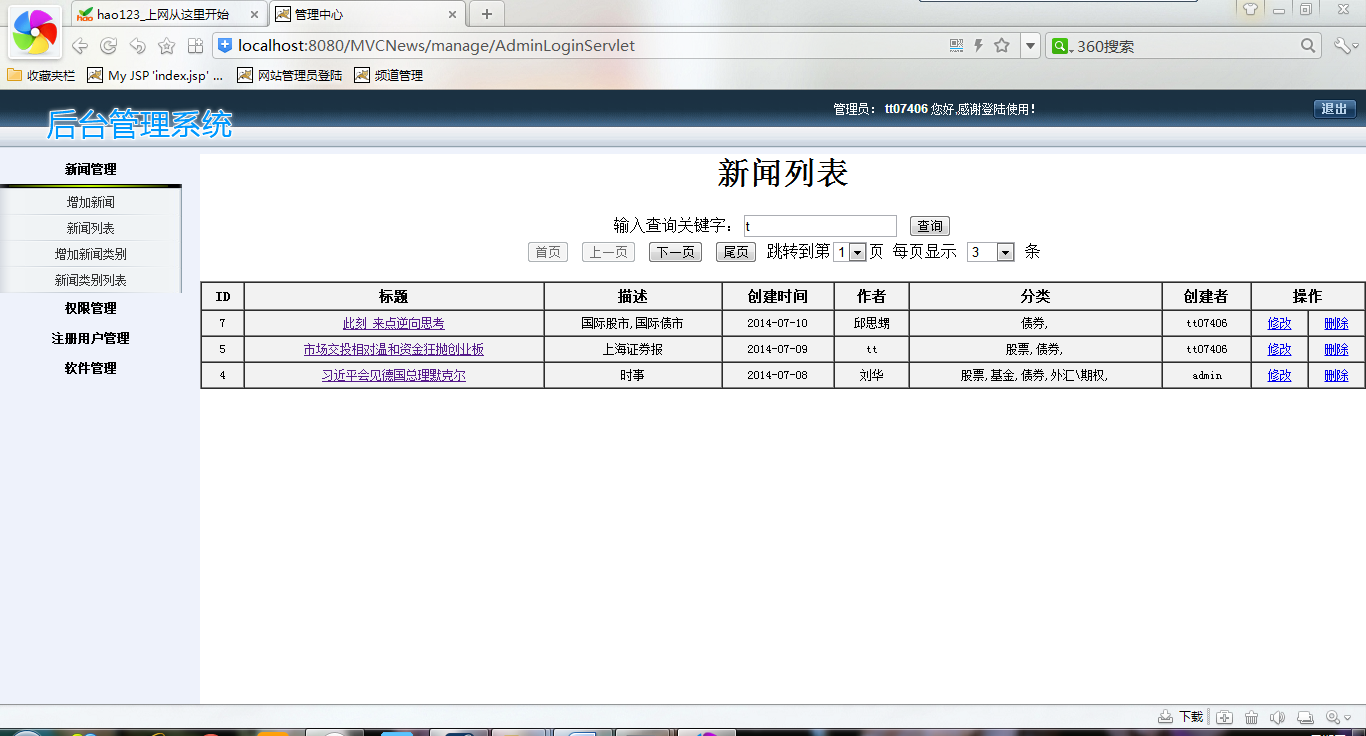


图6.10 翻页和查找

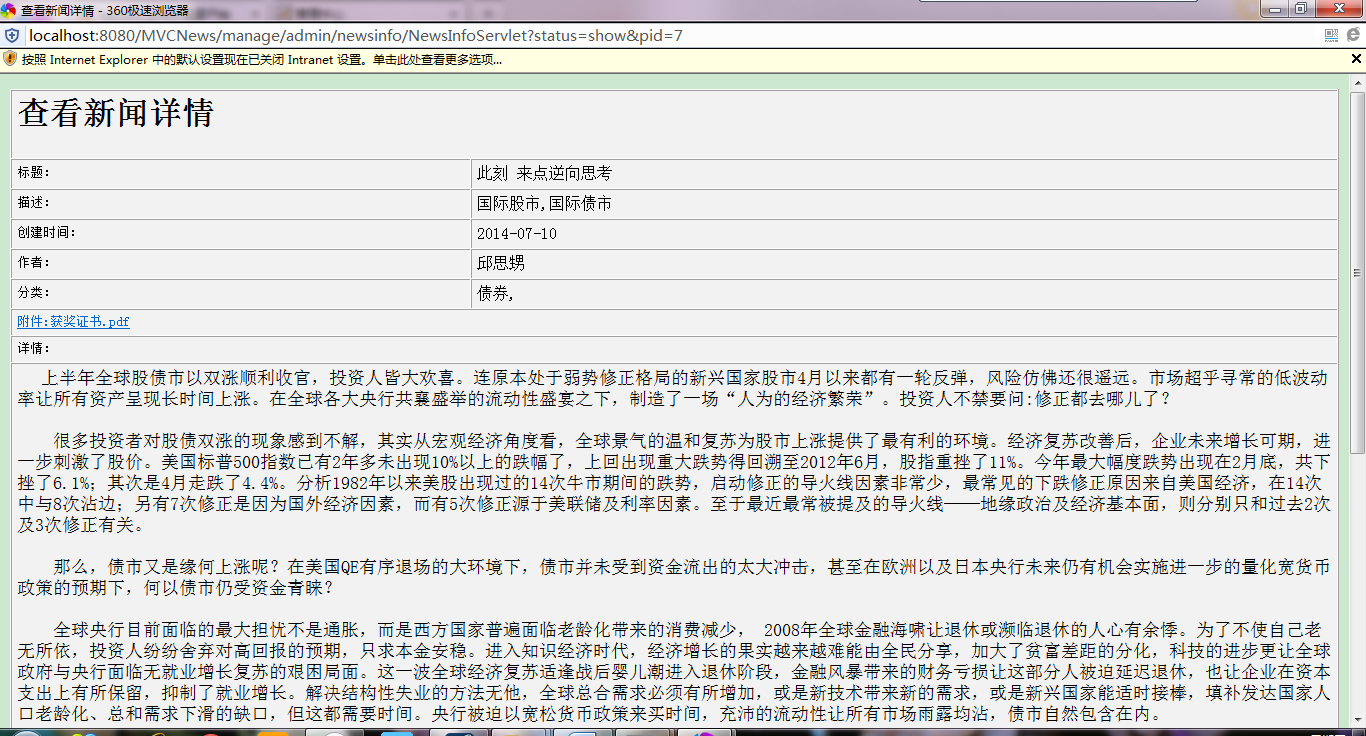


图6.11 新闻详情

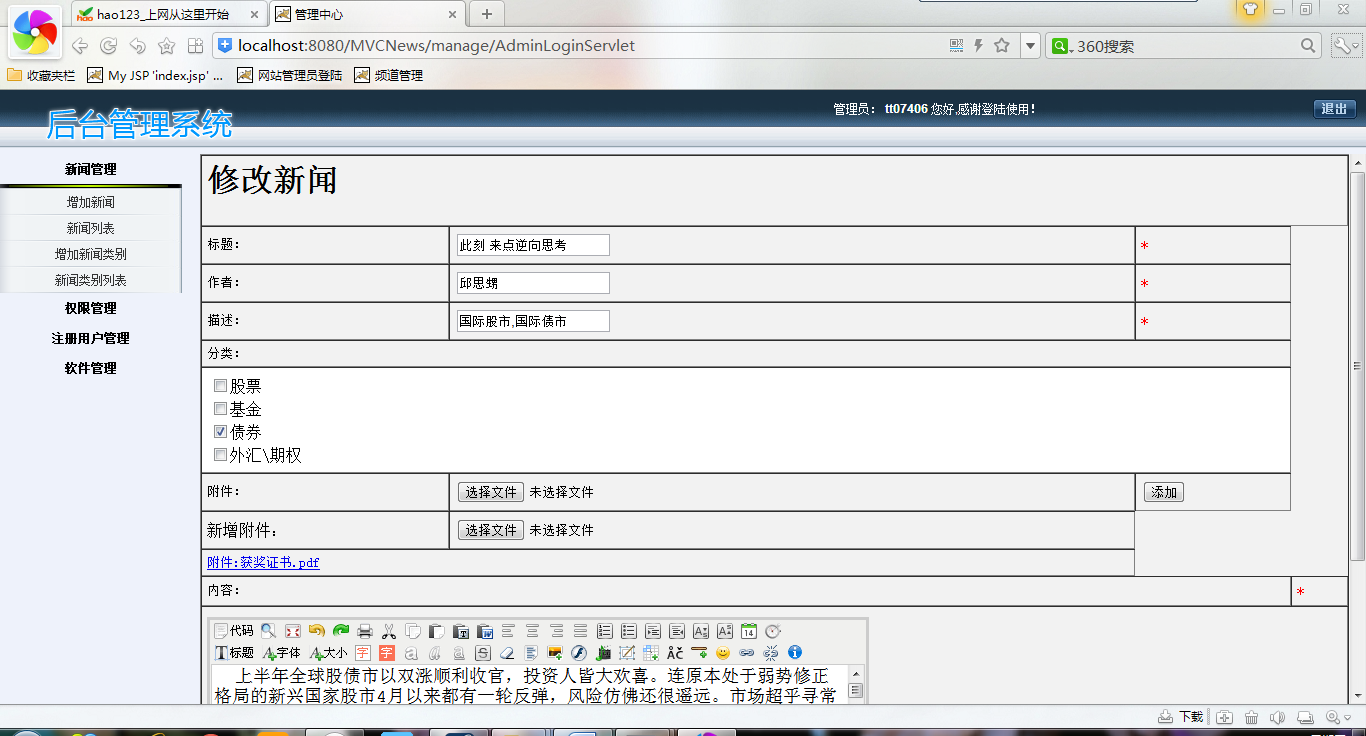
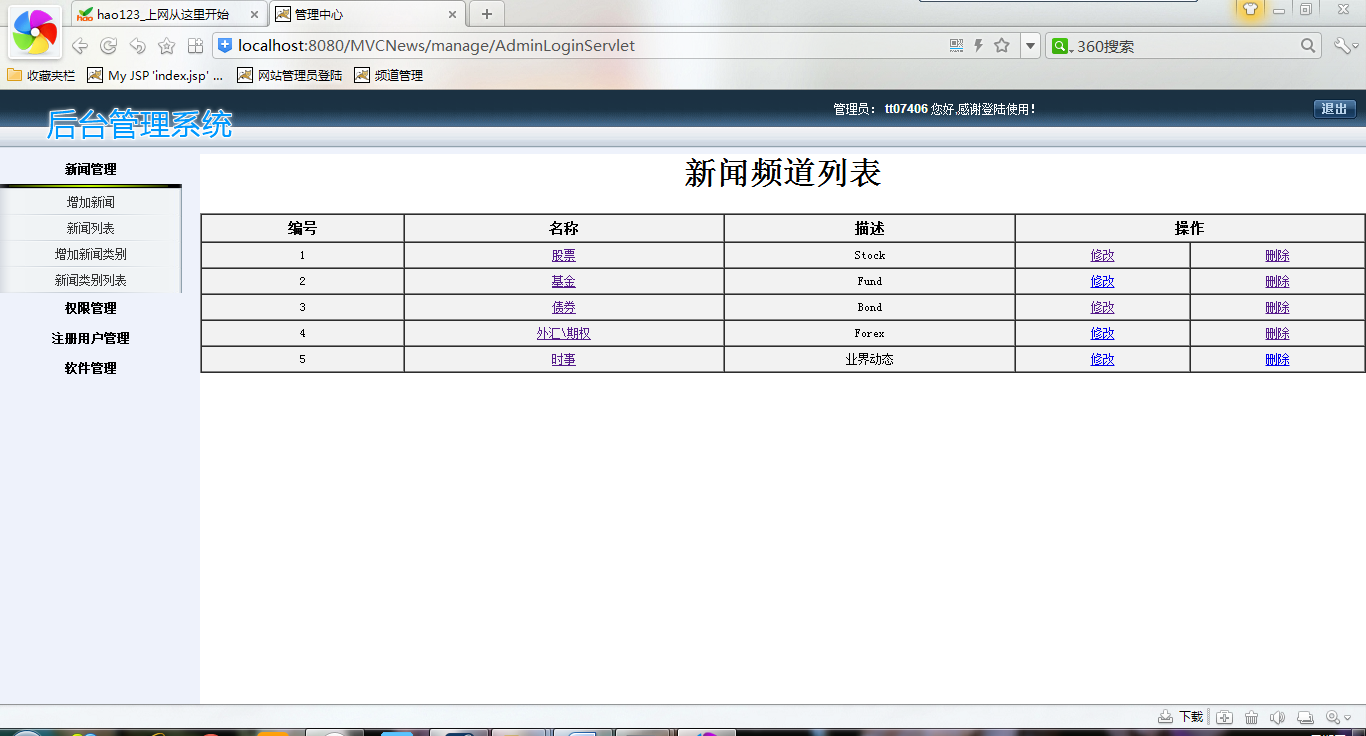


图6.12 修改新闻信息

同样的，可以对新闻类别，管理员进行相应的操作，如下图所示。其中，新闻类别与新闻中的分类项一致，不建议随意增改，修改时也不能修改名称。









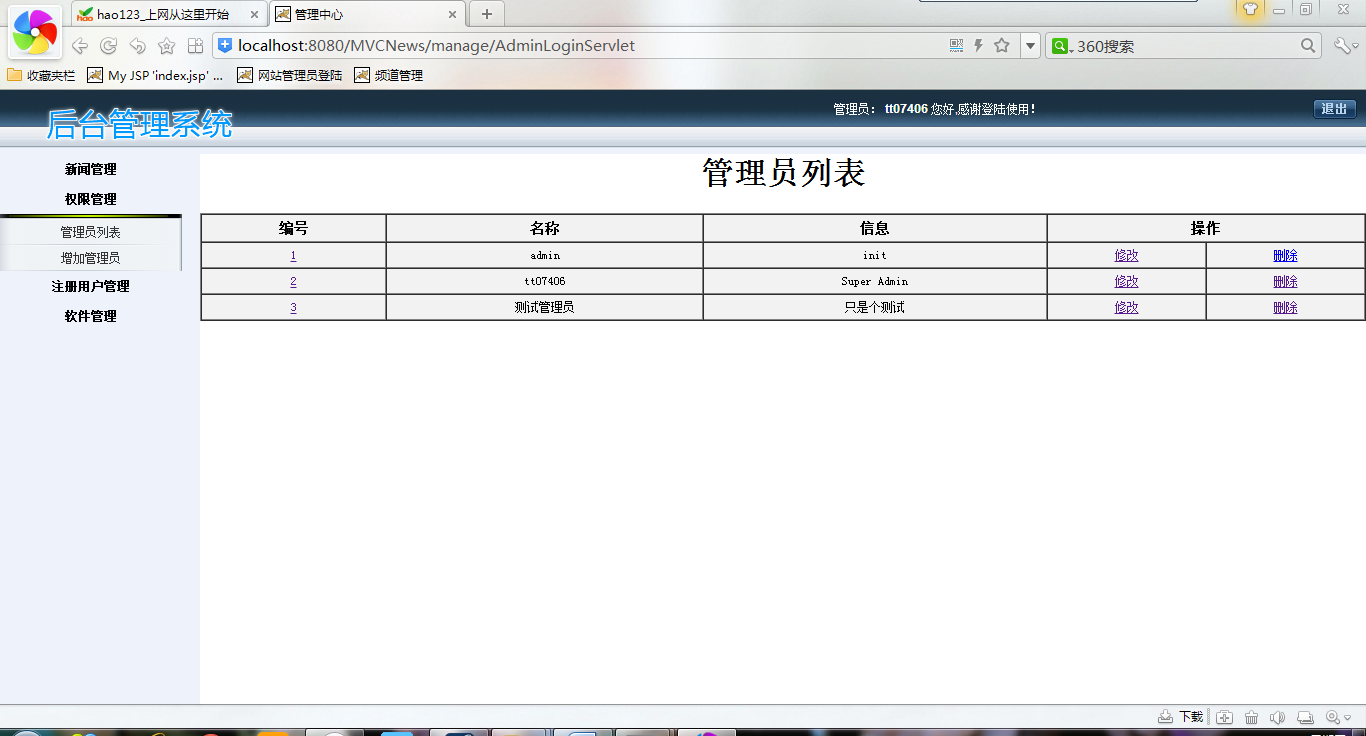


图6.13 新闻类别和管理员操作

对于会员管理，目前只支持展示会员详情、修改会员信息和删除会员操作。

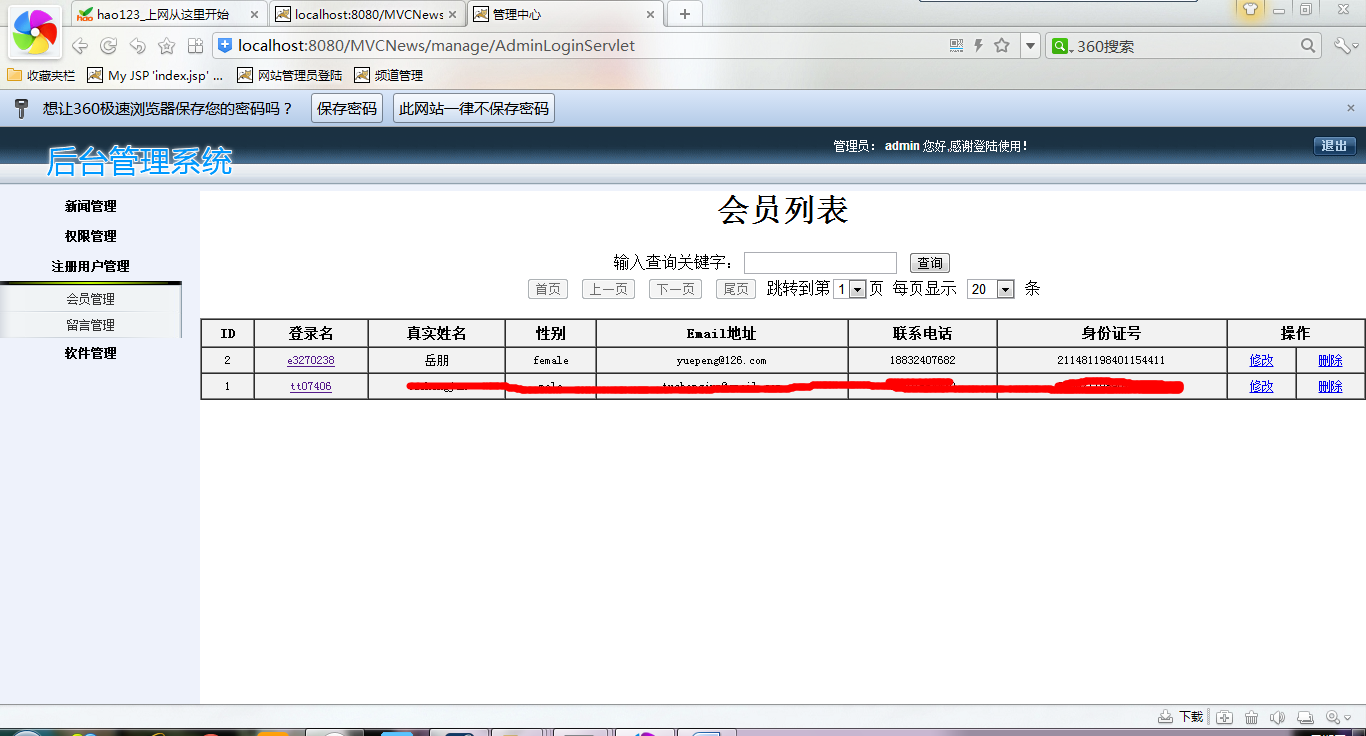


图6.14 会员模块

本系统还可以批量上传软件，软件保存在应用服务器，可以点击链接下载或删除。



图6.15 批量上传软件



图6.16 软件列表



图6.17 软件下载

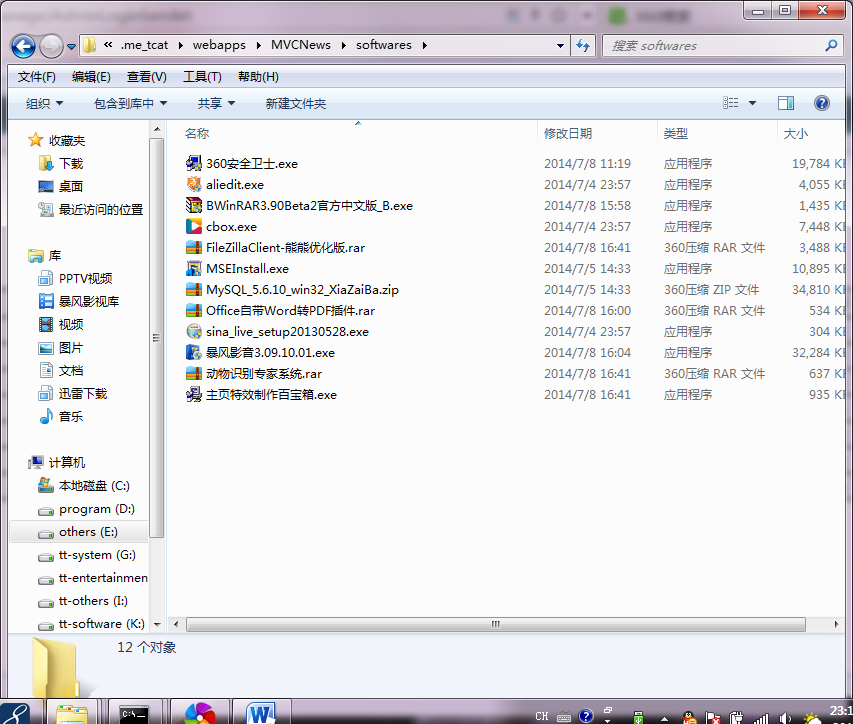


图6.18 软件所在目录