**【摘 要】**同学聚会交友系统是一个用于班级同学增进友谊情感，动态共享，资源共享，师生聊天交流的一个系统。系统建立在Oracle数据库之上，采用JAVA语言和MVC设计模式，同时使用JSP/Servlet、Ajax等技术进行前后台的数据交互，实现在线聊天功能。在系统中，用户可以加入某一个“班级”，并在班级发布或参加班级活动，发布或参与班级话题，通过上传文件到班级空间进行班级分享或在班级空间下载由其他成员上传的文件，或进入聊天室参与班级聊天；或添加好友，查看并评论好友的动态信息、给好友留言；或发表自己的动态，并对好友评论进行回复；或者查看好友给自己的留言，并进行回复等。对系统的需求分析，设计及实现过程做了详细介绍。

**【关键词】**Oracle，JSP/Servlet，Ajax，JAVA，MVC设计模式

Design and Implementation of Classmate Party Dating System

**【****Abstract】**The student dating system is a system for class students to promote friendship, dynamic sharing, resource sharing, teacher and student chat communication. The system is built on the Oracle database, using JAVA language and MVC design pattern, while using JSP / Servlet, Ajax technology for front and back of the data exchange, to achieve online chat. In the system, the user can join a "class" and publish or participate in class activities in the class, publish or participate in class topics, upload files to the class space for class sharing or in the class space to download files uploaded by other members, or Enter the chat room to participate in class chat; or add friends, view and comment on friends of the dynamic information, to a friend message; or publish their own dynamic, and friends comments to reply; or view friends to their message and reply. The requirements analysis, design and implementation of the system are described in detail.  
**【Keywords】**Oracle, JSP / Servlet, Ajax, JAVA, MVC design patterns

目 录

[1 绪论 1](#_Toc19633)

[1.1课题来源 1](#_Toc26176)

[1.2 国内现状 1](#_Toc11507)

[1.3研究意义 2](#_Toc29285)

[2技术介绍 3](#_Toc6747)

[2.1 JAVA开发语言介绍 3](#_Toc10954)

[2.2 JSP技术 4](#_Toc19707)

[2.3 Servlet概述 5](#_Toc26266)

[2.4 JavaBean原理和机制 6](#_Toc26901)

[2.5 AJAX技术 6](#_Toc17361)

[2.6 Maven技术介绍 7](#_Toc23686)

[2.7开发系统 7](#_Toc26973)

[2.8 B/S体系结构介绍 7](#_Toc2277)

[2.9开发环境 8](#_Toc14162)

[3需求分析 9](#_Toc10091)

[3.1可行性分析 9](#_Toc15797)

[3.2安全性分析 9](#_Toc32008)

[3.3性能需求分析 9](#_Toc22187)

[3.4技术可行性 9](#_Toc26722)

[3.5功能需求分析 9](#_Toc26379)

[3.6网络原理图 11](#_Toc24721)

[4数据库设计 12](#_Toc8365)

[4.1数据库概念设计 12](#_Toc29382)

[4.2数据库的表概述 16](#_Toc15327)

[4.3数据库表的结构及说明 16](#_Toc32338)

[5系统实现 25](#_Toc2546)

[5.1系统总体架构及实现 25](#_Toc10650)

[5.2用户登陆 25](#_Toc4282)

[5.3系统首页设计 26](#_Toc8191)

[5.4 我的同学会主页 26](#_Toc8048)

[5.5我的班级 27](#_Toc26571)

[5.6搜索班级 28](#_Toc3940)

[5.7创建班级与加入班级 28](#_Toc22673)

[5.8处理加入班级申请审核 30](#_Toc14159)

[5.9空间动态 31](#_Toc15796)

[5.10发布动态 32](#_Toc3670)

[5.11班级聊天室 33](#_Toc19945)

[5.12基本信息页面 34](#_Toc9208)

[5.13班级活动界面 35](#_Toc10215)

[5.14班级话题界面 36](#_Toc31858)

[5.15好友界面 37](#_Toc17846)

[5.16数据库连接 39](#_Toc7535)

[6系统测试 41](#_Toc9003)

[6.1测试的目的 41](#_Toc13554)

[6.2软件测试的原则 41](#_Toc31128)

[6.3 测试方法 42](#_Toc21730)

[6.4测试过程 42](#_Toc16833)

[6.5测试结果 49](#_Toc20383)

[7总结 50](#_Toc30342)

[后 记 51](#_Toc26242)

[参考文献 52](#_Toc9708)

[附录一： 53](#_Toc16707)

# 1 绪论

## 1.1课题来源

新媒体发展浪潮中，微博已经成为国内最大的社交媒体平台之一。微博用户平稳增长、内容生态丰富多元，社交网络效应凸显，移动化全民性的社交平台逐渐形成，呈现出内容产出视频化、运营垂直化、MCN化等特点。同时，微博也面临过度商业化、低俗炒作等问题。未来，微博将朝着内容分发垂直化、垂直领域内容化、视频领域多元化、矩阵式布局的趋势发展。

在“两微一端”（微博、微信、客户端）等新媒体的引领下，移动社交化的传播格局逐步形成。微博作为以信息发布、互动交流为主的社交媒体平台，兼具社交属性与媒体属性，在拥有庞大用户的基础之上，成为巨大的流量入口。

## 1.2国内现状

目前国内外无数优秀的微博系统开始涌现。它们允许用户以简短文字随时随地更新自己的状态，每条信息的长度都在140字以内，支持图片、音频、视频等多媒体的出版，每个用户既是微内容的创造者也是微内容的传播者和分享者，极大得拉低了用户的创作门槛，这140字的内容，让每个人都成了莎士比亚。

（一）用户持续增长，迈向移动化全民性社交平台

《2017年微博用户发展报告》显示，截至2017年9月，微博月活跃人数共3.76亿，与2016年同期相比增长27%；日活跃用户达1.65亿，较2016年同期增长25%。微博月活跃用户中，30岁以下用户超过八成，是微博的主力人群；在性别上，男性用户占比56.3%，女性用户占比43.7%。来自三四线城市的用户占微博月活跃用户的52.6%，微博的区域覆盖进一步下沉。活跃用户规模呈现持续稳步增长的态势，性别比例趋向平衡，区域覆盖范围进一步加强，目前微博正朝着建设全民性社交媒体平台的方向迈进。

从用户行为看，微博月活用户已经有92%来自移动端；月登录天数在15天以上的高黏性用户比例最高；微博用户兴趣主要集中在明星、美女帅哥、动漫等泛娱乐大众领域；同时，文学、情感、股票等也是微博用户的主要兴趣标签[1]。

微博用户的使用习惯趋向于移动化，移动消费时间的碎片化成为普遍现象；高黏性用户成为微博用户的构成主体；用户兴趣分布广泛，同时也形成群族化的个性化需求，泛娱乐领域仍然是微博活跃的主要场所。

（二）多形式博文齐发，内容生态丰富多元

微博作为开放的社交媒体，庞大的用户群是其发展依托，而内容则是其发展的基础与核心。近年来，微博注重内容建设，丰富多元化的内容形态进一步提升了用户体验。

目前，图文类博文仍然是微博用户最主要的发布形式，与此同时，包含链接、视频及音乐等多形式博文的占比得到全面提升。2016年，微博就引入直播进一步强化在短视频领域的覆盖传播优势。2017年第三季度微博视频播放量同比增长175%，高清视频发布量占比为30%。目前微博正继续加强视频领域的布局，不断优化视频的内容品质和观看体验。与此同时，微博与各音乐平台加大合作力度，不断提升音乐的分享体验。

头条文章作为微博最新的长文产品，与普通微博相比，在内容上更丰富、详细，是碎片化内容的弥补与增强，一定程度上能够满足和加深微博用户深度阅读的需求。新浪数据中心报告显示，2017年1—9月微博头条文章发布量同比增长90%。以信息流大卡片出现的头条文章借助于微博传播优势，能够以更快更准的方式传播触达到更多用户。微博用户不仅能对头条文章进行转发、评论、点赞、收藏，还可以通过打赏、付费阅读等功能，支持鼓励原创作者持续产出更多优质内容。

为构建具有影响力的短视频创作和消费平台，微博故事于2017年4月上线。微博故事鼓励普通用户创作，分享短视频内容。微博故事创作、发布门槛低，获得了年轻普通女性用户的青睐。

（三）聚焦用户需求，社交网络效应凸显

近年来，微博的活跃用户持续稳步增长，高黏性用户占比不断提高，性别比例趋向平衡，区域覆盖范围进一步加强。庞大的用户规模，使微博能够获得内容“首发优势”。微博关注方式所形成的弱关系社会网络使微博的传播机制更接近于大众传播，也为裂变式传播奠定基础，有利于推动热点事件的形成和舆论的发酵。因此，内容发布者为追求最大的传播声量和热度，会选择微博作为发声平台。

庞大活跃的用户数量成为强大的内容生态的基础。为争夺用户注意力、抢占用户心智，微博聚焦用户需求，加大在内容运营方面的投入力度。无论是在视频领域的布局，还是进行垂直化运营，加大扶持MCN机构，都是在为方便用户创造、分享内容，获取信息创造条件。随着用户需求得到满足，用户数量会进一步增加，黏性会进一步提高。由此，微博的网络价值得以实现几何级数增长，社交网络效应得以凸显。

## 1.3研究意义

随着计算机技术的飞速发展，大数据时代已经到来，越来越多的移动端应用开始瞄准了用户的碎片化时间（手机游戏，短视频应用等），微博的出现尤其影响了人们的日常社交方式。

因此，微博系统的重要性不言而喻，一个稳定，高效，安全的微博系统，方便了用户的社交，加深了用户的交流，保护了用户的隐私，还优化了从发布内容到浏览信息的所有细节，降低了用户的上手操作门槛，拉近了每一个用户之间的距离。

# 2开发环境

## 2.1 硬件

MacBook Pro (13-inch, 2018)

系统macOS Catalina 10.15.3

CPU 2.3 GHz 四核Intel Core i5

内存 8 GB 2133 MHz LPDDR3

磁盘 Macintosh HD

腾讯云主机 (标准型S2 1Mbps)

系统 Ubuntu 16.04.1 LTS

CPU 1核

内存 2GB

磁盘 高性能云硬盘。

## 2.2 软件

JDK 1.8.0\_231

Tomcat 8.5.50

MySQL 5.7

IntelliJ IDEA 2019.3.2

Navicat Premium 15.0.8

Spring 4.2.23

MyBatis 3.3.0

# 2技术路线

## 2.1 技术栈

后端：

Java

Spring

SpringMVC

Tomcat

MySQL

log4J

MyBatis

Maven

Linux

WebSocket

HTTP

前端：

HTML5

CSS

JavaScript

Jsp

jquery

bootstrap

## 2.1 项目依赖

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>5.1.4</version>

</dependency>

<dependency>

<groupId>com.github.jsqlparser</groupId>

<artifactId>jsqlparser</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-all</artifactId>

<version>1.9.5</version>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.8</version>

</dependency>

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest-core</artifactId>

<version>1.3</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>5.4.1.Final</version>

</dependency>

<!-- https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.3.3</version>

</dependency>

<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>1.3.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-log4j12 -->

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.6</version>

</dependency>

<dependency>

<groupId>net.coobird</groupId>

<artifactId>thumbnailator</artifactId>

<version>0.4.7</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.12</version>

</dependency>

<dependency>

<groupId>com.drewnoakes</groupId>

<artifactId>metadata-extractor</artifactId>

<version>2.9.1</version>

</dependency>

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-api</artifactId>

<version>7.0</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.3.23.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.mchange.c3p0/com.springsource.com.mchange.v2.c3p0 -->

<!-- https://mvnrepository.com/artifact/com.mchange/c3p0 -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5.4</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.23.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-websocket -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-websocket</artifactId>

<version>4.3.23.RELEASE</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.11</version>

</dependency>

<!--jstl-->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>1.1.2</version>

</dependency>

<!--mybatis-->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.3.0</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.2.3</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->

<!-- https://mvnrepository.com/artifact/org.springframework/org.springframework.jdbc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.3.23.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>4.3.23.RELEASE</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<!--确保版本与其他spring开头的依赖相同 -->

<version>4.3.23.RELEASE</version>

</dependency>

<!-- 打包时报错加的https://mvnrepository.com/artifact/javax.activation/activation -->

<dependency>

<groupId>javax.activation</groupId>

<artifactId>activation</artifactId>

<version>1.1</version>

</dependency>

## 2.1 项目结构

arvinclub-blog

├── README.md

├── blog-model

│   ├── pom.xml

│   ├── src

│   │   ├── main

│   │   │   └── java

│   │   │   └── com

│   │   │   └── arvinclub

│   │   │   └── model

│   │   │   ├── entity

│   │   │   │   ├── Blog.java

│   │   │   │   ├── Comment.java

│   │   │   │   └── User.java

│   │   │   └── util

│   │   │   ├── BASE64DecodedMultipartFile.java

│   │   │   ├── ImageSizeUtil.java

│   │   │   ├── MutedException.java

│   │   │   └── ToolsUtil.java

│   │   └── test

│   │   └── sql.txt

│   └── target

│   ├── arvinclub-blog.jar

│   ├── classes

│   ├── generated-sources

│   ├── maven-archiver

│   └── maven-status

│

│

├── blog-service

│   ├── pom.xml

│   ├── src

│   │   ├── main

│   │   │   ├── java

│   │   │   │   └── com

│   │   │   │   └── arvinclub

│   │   │   │   └── service

│   │   │   │   ├── config

│   │   │   │   │   ├── GetHttpSessionConfigurator.java

│   │   │   │   │   ├── RootConfig.java

│   │   │   │   │   └── WebSocket.java

│   │   │   │   ├── dao

│   │   │   │   │   ├── BlogDao.java

│   │   │   │   │   ├── CommentDao.java

│   │   │   │   │   └── UserDao.java

│   │   │   │   └── service

│   │   │   │   ├── BlogService.java

│   │   │   │   ├── CommentService.java

│   │   │   │   ├── UserService.java

│   │   │   │   └── impl

│   │   │   │   ├── BlogServiceImpl.java

│   │   │   │   ├── CommentServiceImpl.java

│   │   │   │   └── UserServiceImpl.java

│   │   │   └── resources

│   │   │   ├── database.properties

│   │   │   ├── log4j.properties

│   │   │   └── mapper

│   │   │   ├── blogMapper.xml

│   │   │   ├── commentMapper.xml

│   │   │   └── userMapper.xml

│   │   └── test

│   │   └── java

│   │   └── com

│   │   └── arvinclub

│   │   └── service

│   │   └── MyTest.java

│   └── target

│   ├── arvinclub-blog.jar

│   ├── classes

│   ├── generated-sources

│   ├── generated-test-sources

│   ├── maven-archiver

│   ├── maven-status

│   ├── surefire-reports

│   └── test-classes

│

│

├── blog-web

│   ├── pom.xml

│   ├── src

│   │   └── main

│   │   ├── java

│   │   │   └── com

│   │   │   └── arvinclub

│   │   │   └── web

│   │   │   ├── config

│   │   │   │   ├── BlogWebAppInitializer.java

│   │   │   │   └── WebConfig.java

│   │   │   ├── controller

│   │   │   │   ├── admin

│   │   │   │   │   └── AdminController.java

│   │   │   │   ├── blog

│   │   │   │   │   ├── BlogsController.java

│   │   │   │   │   └── CommentController.java

│   │   │   │   ├── error

│   │   │   │   │   └── ErrorController.java

│   │   │   │   ├── fun

│   │   │   │   │   ├── PageController.java

│   │   │   │   │   ├── RredirectController.java

│   │   │   │   │   └── SearchController.java

│   │   │   │   ├── other

│   │   │   │   │   └── DownloadController.java

│   │   │   │   └── user

│   │   │   │   ├── AttentionController.java

│   │   │   │   ├── LoginController.java

│   │   │   │   └── RegisteController.java

│   │   │   └── filter

│   │   │   ├── AdminFilter.java

│   │   │   ├── CharactorFilter.java

│   │   │   └── userFilter.java

│   │   ├── resources

│   │   │   └── static

│   │   │   ├── assets

│   │   │   │   ├── css

│   │   │   │   ├── fonts

│   │   │   │   ├── images

│   │   │   │   └── js

│   │   │   ├── css

│   │   │   ├── favicon.ico

│   │   │   ├── file

│   │   │   │   └── images

│   │   │   ├── fonts

│   │   │   ├── images

│   │   │   ├── js

│   │   │   ├── lib

│   │   │   │   └── layui

│   │   │   │   ├── css

│   │   │   │   ├── font

│   │   │   │   ├── images

│   │   │   │   ├── lay

│   │   │   ├── prepros-6.config

│   │   │   └── vendor

│   │   │   ├── bootstrap

│   │   │   │   ├── css

│   │   │   │   └── js

│   │   │   └── jquery

│   │   └── webapp

│   │   └── WEB-INF

│   │   ├── admin

│   │   │   └── list.jsp

│   │   ├── attention.jsp

│   │   ├── base.jsp

│   │   ├── blogs.jsp

│   │   ├── chat.jsp

│   │   ├── detail.jsp

│   │   ├── error.jsp

│   │   ├── list.jsp

│   │   ├── login.jsp

│   │   ├── page.jsp

│   │   ├── registe.jsp

│   │   └── search.jsp

│   └── target

│   ├── blog

│   │   ├── META-INF

│   │   └── WEB-INF

│   ├── blog.war

│   ├── classes

│   ├── generated-sources

│   └── maven-archiver

│

│

│

│

└── pom.xml

# 2技术介绍

## 2.1 JAVA开发语言介绍

Java语言作为1995年Sun公司推出的跨系统性的编程语言，Java是一个广泛使用的网络编程语言 ,它是一种新的计算概念。

首先 ,作为一种程序设计语言 ,它简单、面向对象、不依赖于机器的结构、具有可移植性、鲁棒性、安全性、并且提供了并发的机制、具有很高的性能。其次 ,它最大限度地利用了网络 ,Java的小应用程序 (applet)可在网络上传输而不受 CPU和环境的限制。另外 ,Java还提供了丰富的类库 ,使程序设计者可以很方便地建立自己的系统。

下面我们分别从这三个方面来讨论 Java的特点 ,然后通过把 Java与 C,C++相比进一步指出它所具有的优点。

一、 Java语言

Java语言有下面一些特点 :简单、面向对象、分布式、解释执行、鲁棒、安全、体系结构中立、可移植、高性能、多线程以及动态性。

1.简单性

Java语言是一种面向对象的语言 ,它通过提供最基本的方法来完成指定的任务 ,只需理解一些基本的概念 ,就可以用它编写出适合于各种情况的应用程序。 Java略去了运算符重载、多重继承等模糊的概念 ,并且通过实现自动垃圾收集大大简化了程序设计者的内存管理工作。另外 ,Java也适合于在小型机上运行 ,它的基本解释器及类的支持只有 40KB左右 ,加上标准类库和线程的支持也只有 215KB左右。库和线程的支持也只有 215KB左右。

2.面向对象

Java语言的设计集中于对象及其接口 ,它提供了简单的类机制以及动态的接口模型。对象中封装了它的状态变量以及相应的方法 ,实现了模块化和信息隐藏 ;而类则提供了一类对象的原型 ,并且通过继承机制 ,子类可以使用父类所提供的方法 ,实现了代码的复用。

3.分布性

Java是面向网络的语言。通过它提供的类库可以处理 TCP/IP协议 ,用户可以通过 URL地址在网络上很方便地访问其它对象。

4.鲁棒性 Java在编译和运行程序时 ,都要对可能出现的问题进行检查 ,以消除错误的产生。它提供自动垃圾收集来进行内存管理 ,防止程序员在管理内存时容易产生的错误。通过集成的面向对象的例外处理机制 ,在编译时,Java提示出可能出现但未被处理的例外 ,帮助程序员正确地进行选择以防止系统的崩溃。另外,Java在编译时还可捕获类型声明中的许多常见错误 ,防止动态运行时不匹配问题的出现。

5.安全性

用于网络、分布环境下的 Java必须要防止病毒的入侵。 Java不支持指针,一切对内存的访问都必须通过对象的实例变量来实现 ,这样就防止程序员使用"特洛伊 "木马等欺骗手段访问对象的私有成员 ,同时也避免了指针操作中容易产生的错误。

6.体系结构中立

Java解释器生成与体系结构无关的字节码指令 ,只要安装了 Java运行时系统 ,Java程序就可在任意的处理器上运行。这些字节码指令对应于 Java虚拟机中的表示 ,Java解释器得到字节码后 ,对它进行转换 ,使之能够在不同的平台运行。

7.可移植性

与平台无关的特性使 Java程序可以方便地被移植到网络上的不同机器。同时 ,Java的类库中也实现了与不同平台的接口 ,使这些类库可以移植。另外,Java编译器是由 Java语言实现的 ,Java运行时系统由标准 C实现 ,这使得Java系统本身也具有可移植性。

8.解释执行

Java解释器直接对 Java字节码进行解释执行。字节码本身携带了许多编译时信息 ,使得连接过程更加简单。

9.高性能

和其它解释执行的语言如 BASIC、 TCL不同 ,Java字节码的设计使之能很容易地直接转换成对应于特定CPU的机器码 ,从而得到较高的性能。

10.多线程

多线程机制使应用程序能够并行执行 ,而且同步机制保证了对共享数据的正确操作。通过使用 多线程 ,程序设计者可以分别用不同的线程完成特定的行为 ,而不需要采用全局的事件循环机制 ,这样就很容易地实现网络上的实时交互行为。

11.动态性

Java的设计使它适合于一个不断发展的环境。在类库中可以自由地加入新的方法和实例变量而不会影响用户程序的执行。并且 Java通过接口来支持多重继承 ,使之比严格的类继承具有更灵活的方式和扩展性。

## 2.1 Maven简介

1. 历史

[Maven](https://baike.baidu.com/item/Maven)项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的[项目管理工具](https://baike.baidu.com/item/项目管理工具/6854630)软件。

Maven 除了以程序构建能力为特色之外，还提供高级项目管理工具。由于 Maven 的缺省构建规则有较高的可重用性，所以常常用两三行 Maven 构建脚本就可以构建简单的项目。由于 Maven 的面向项目的方法，许多 Apache Jakarta 项目发文时使用 Maven，而且公司项目采用 Maven 的比例在持续增长。

Maven设计之初， 是为了简化Jakarta Turbine项目的建设。 在几个项目， 每个项目包含了不同的Ant构建文件。 JAR检查到CVS。 Apache组织开发Maven可以建立多个项目， 发布项目信息， 项目部署， 在几个项目中JAR文件提供团队合作和帮助。

Maven的经历了Maven-> Maven2 -> Maven3的发展

2. 国内外现状

国内外都有无数知名项目通过maven构建，并托管在GitHub平台

## 2.1 Spring简介

1. 历史

2002 年 10 月，Rod Johnson 撰写了一本名为 Expert One-on-One J2EE 设计和开发的书。本书由 Wrox出版，介绍了当时 Java 企业应用程序开发的情况，并指出了 Java EE 和 EJB 组件框架中的存在的一些主要缺陷。在这本书中，他提出了一个基于普通 Java 类和依赖注入的更简单的解决方案。

在本书发布后不久，开发者 Juergen Hoeller 和 Yann Caroff 说服 Rod Johnson 创建一个基于基础结构代码的开源项目。Rod，Juergen 和 Yann 于 2003 年 2 月左右开始合作开发该项目 。Yann 为新框架创造了“Spring”的名字。Yann Caroff 在早期离开了团队，Rod Johnson 在 2012 年离开，Juergen Hoeller 仍然是 Spring 开发团队的积极成员。

自 2004 年 1.0 版本发布以来，Spring 框架迅速发展。Spring 2.0 于 2006 年 10 月发布，到那时，Spring的下载量超过了 100 万。Spring 2.0 具有可扩展的 XML 配置功能，用于简化 XML 配置，支持 Java 5，额外的 IoC 容器扩展点，支持动态语言。

在 Rod 领导下管理 Interface21 项目于 2007 年 11 月更名为 SpringSource。同时发布了 Spring 2.5。Spring 2.5 中的主要新功能包括支持 Java 6 / Java EE 5，支持注释配置，classpath 中的组件自动检测和兼容 OSGi 的 bundle。

2007 年，SpringSource 从基准资本获得了 A 轮融资（1000万美元）。SpringSource 在此期间收购了多家公司，如Hyperic，G2One 等。2009年8月，SpringSource 以 4.2 亿美元被 VMWare 收购。SpringSource 在几周内收购了云代工厂，这是一家云 PaaS 提供商。2015 年，云代工厂转型成了非营利云代工厂。

2009 年 12 月，Spring 3.0 发布。Spring 3.0 具有许多重要特性，如重组模块系统，支持 Spring 表达式语言，基于 Java 的 bean 配置（JavaConfig），支持嵌入式数据库（如 HSQL，H2 和 Derby），模型验证/ REST 支持和对 Java EE 的支持。

2011 年和 2012 年发布了许多 3.x 系列的小版本。2012 年 7 月，Rod Johnson 离开了团队。2013 年 4月，VMware 和 EMC 通过 GE 投资创建了一家名为 Pivotal 的合资企业。所有的 Spring 应用项目都转移到了 Pivotal。

2013 年 12 月，Pivotal 宣布发布 Spring 框架 4.0。Spring 4.0 是 Spring 框架的一大进步，它包含了对Java 8 的全面支持，更高的第三方库依赖性（groovy 1.8+，ehcache 2.1+，hibernate 3.6+等），Java EE 7 支持，groovy DSL for bean 定义，对 websockets 的支持以及对泛型类型的支持作为注入 bean 的限定符。

2014 年至 2017 年期间发布了许多 Spring 框架 4.xx 系列版本。

Spring 5.0 GA版本于2017年9月28日发布。Spring 5.0开始支持JDK 8和Java EE 7，同时兼容JDK9。全面支持Servlet 3.1，还引入了一个全新的模块Spring WebFlux用于替代老话的 spring-webmvc；对Kotlin也有了更好的支持。

本项目中使用的版本为Spring 4

Spring 4.x新特性：

Spring 4.x全面支持Java 8.0，支持Lambda表达式的使用，提供了对@Scheduled和@PropertySource重复注解的支持，提供了空指针终结者Optional，对核心容器进行增加：支持泛型的依赖注入、Map的依赖注入、Lazy延迟依赖的注入、List注入、Condition条件注解注入、对CGLib动态代理类进行了增强。

Spring 4.x还支持了基于Groovy DSL的配置，提高Bean配置的灵活性。

Spring 4.x开始，Spring MVC基于Servlet 3.0 开发，并且为了方便Restful开发，引入了新的RestController注解器注解，同时还增加了一个AsyncRestTemplate支持Rest客户端的异步无阻塞请求。

Spring框架为任何类型的部署平台上的基于Java的现代企业应用程序提供了全面的编程和配置模型。

Spring的一个关键元素是在应用程序级别的基础架构支持：Spring专注于企业应用程序的“管道”，以便团队可以专注于应用程序级别的业务逻辑，而不必与特定的部署环境建立不必要的联系。

支持政策和迁移

特征

[核心技术]依赖项注入，事件，资源，i18n，验证，数据绑定，类型转换，SpEL，AOP。

[测试]模拟对象，TestContext框架，Spring MVC测试，`WebTestClient`。

[数据访问]事务，DAO支持，JDBC，ORM，封送XML。

[Spring MVC]和 [Spring WebFlux] Web框架。

[集成]远程处理，JMS，JCA，JMX，电子邮件，任务，调度，缓存。

[语言] Kotlin，Groovy，动态语言。

1. 国内外现状

国内外都有无数知名Java项目通过maven构建，并托管在GitHub平台

IOC

本章介绍了控制反转（IoC）原理的Spring框架实现。IoC也称为依赖注入（DI）。在此过程中，对象仅通过构造函数参数，工厂方法的参数或在构造或从工厂方法返回后在对象实例上设置的属性来定义其依赖项（即，与它们一起使用的其他对象） 。然后，容器在创建bean时注入那些依赖项。此过程从根本上讲是通过使用类的直接构造或诸如服务定位器模式之类的机制来控制其依赖项的实例化或位置的bean本身的逆过程（因此称为Control Inversion）。

在org.springframework.beans和org.springframework.context包是Spring框架的IoC容器的基础。该 [BeanFactory] 界面提供了一种高级配置机制，能够管理任何类型的对象。 [ApplicationContext]是的子接口BeanFactory。它增加了：

与Spring的AOP功能轻松集成

消息资源处理（用于国际化）

活动发布

应用层特定的上下文，例如WebApplicationContext 用于Web应用程序中的。

简而言之，BeanFactory提供了配置框架和基本功能，并`ApplicationContext`增加了更多针对企业的功能。该ApplicationContext是对一个完整的超集`BeanFactory`，并在Spring的IoC容器的描述本章独占使用。有关使用的详细信息`BeanFactory`，而不是ApplicationContext,看到 [的BeanFactory]。

在Spring中，构成应用程序主干并由Spring IoC容器管理的对象称为bean。Bean是由Spring IoC容器实例化，组装和以其他方式管理的对象。否则，bean仅仅是应用程序中许多对象之一。Bean及其之间的依赖关系反映在容器使用的配置元数据中。

1.2 容器概述

该org.springframework.context.ApplicationContext接口代表Spring IoC容器，并负责实例化，配置和组装Bean。容器通过读取配置元数据来获取有关要实例化，配置和组装哪些对象的指令。配置元数据以XML，Java批注或Java代码表示。它使您能够表达组成应用程序的对象以及这些对象之间的丰富相互依赖关系。

ApplicationContext`Spring提供了该接口的几种实现。在独立应用程序中，通常创建[ClassPathXmlApplicationContext]或的实例 [FileSystemXmlApplicationContext]。尽管XML是定义配置元数据的传统格式，但是您可以通过提供少量XML配置来声明性地启用对这些其他元数据格式的支持，从而指示容器将Java注释或代码用作元数据格式。

在大多数应用场景中，不需要显式的用户代码来实例化一个Spring IoC容器的一个或多个实例。例如，在Web应用程序场景中，应用程序文件中的简单八行（约）样板Web描述符XML web.xml通常就足够了。如果使用 [Spring Tool Suite]（基于Eclipse的开发环境），则只需单击几次鼠标或击键即可轻松创建此样板配置。

下图显示了Spring的工作原理的高级视图。您的应用程序类与配置元数据结合在一起，因此，在ApplicationContext创建和初始化之后，您将拥有一个完全配置且可执行的系统或应用程序。

AOP

面向方面的编程（AOP）通过提供另一种思考程序结构的方式来补充面向对象的编程（OOP）。OOP中模块化的关键单元是类，而在AOP中模块化是方面。方面使关注点（例如事务管理）的模块化跨越了多个类型和对象。（这种关注在AOP文献中通常被称为“跨领域”关注。）

Spring的关键组件之一是AOP框架。虽然Spring IoC容器不依赖于AOP（这意味着您不需要使用AOP），但AOP是对Spring IoC的补充，以提供功能非常强大的中间件解决方案。

具有AspectJ切入点的Spring AOP

Spring通过使用[基于模式的方法]或[@AspectJ批注样式，]提供了编写自定义方面的简单而强大的方法 。这两种样式都提供了完全类型化的建议，并使用了AspectJ切入点语言，同时仍然使用Spring AOP进行编织。

本章讨论基于架构和基于@AspectJ的AOP支持。

AOP在Spring Framework中用于：

提供声明式企业服务。这种服务最重要的是 [声明式事务管理]。

让用户实现自定义方面，并通过AOP补充其对OOP的使用。

SpringMVC

Spring Web MVC是基于Servlet API构建的原始Web框架，并且从一开始就已包含在Spring框架中。正式名称“Spring Web MVC”来自其源模块（[spring-webmvc]）的名称，但它通常被称为“ Spring MVC”。

与Spring Web MVC并行，Spring Framework 5.0引入了一个反应式堆栈Web框架，其名称“ Spring WebFlux”也基于其源模块（[spring-webflux]）。

## 2.1 MyBatis简介

MyBatis 是支持定制化[ SQL](https://www.w3cschool.cn/sql/)、存储过程以及高级映射的优秀的持久层框架。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以对配置和原生Map使用简单的 XML 或注解，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。

MyBatis SQL映射器框架使将关系数据库与面向对象的应用程序一起使用变得更加容易。MyBatis使用XML描述符或注释将对象与存储过程或SQL语句耦合。相对于对象关系映射工具，简单性是MyBatis数据映射器的最大优势。

优点：

简单易学：本身就很小且简单。没有任何第三方依赖，最简单安装只要两个jar文件+配置几个sql映射文件易于学习，易于使用，通过文档和源代码，可以比较完全的掌握它的设计思路和实现。

灵活：mybatis不会对应用程序或者数据库的现有设计强加任何影响。 sql写在xml里，便于统一管理和优化。通过sql基本上可以实现我们不使用数据访问框架可以实现的所有功能，或许更多。

解除sql与程序代码的耦合：通过提供DAL层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元测试。sql和代码的分离，提高了可维护性。

提供映射标签，支持对象与数据库的orm字段关系映射

提供对象关系映射标签，支持对象关系组建维护

提供xml标签，支持编写动态sql。

缺点：

编写SQL语句时工作量很大，尤其是字段多、关联表多时，更是如此。

SQL语句依赖于数据库，导致数据库移植性差，不能更换数据库。

框架还是比较简陋，功能尚有缺失，虽然简化了数据绑定代码，但是整个底层数据库查询实际还是要自己写的，工作量也比较大，而且不太容易适应快速数据库修改。

二级缓存机制不佳

MyBatis的功能架构：

我们把Mybatis的功能架构分为三层：

1. API接口层：提供给外部使用的接口API，开发人员通过这些本地API来操纵数据库。接口层一接收到调用请求就会调用数据处理层来完成具体的数据处理。

2. 数据处理层：负责具体的SQL查找、SQL解析、SQL执行和执行结果映射处理等。它主要的目的是根据调用的请求完成一次数据库操作。

3. 基础支撑层：负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为最基础的组件。为上层的数据处理层提供最基础的支撑。

Mybatis的使用

要使用 MyBatis， 只需将 mybatis-x.x.x.jar 文件置于 classpath 中即可。

如果使用 Maven 来构建项目，则需将下面的 dependency 代码置于 pom.xml 文件中：

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>x.x.x</version>

</dependency>

## 2.1 Tomcat简介与搭建

简介

Apache Tomcat 支持了Java Servlet，JavaServer Page，Java Expression Language和Java的WebSocket技术的一个开源Web容器。Java Servlet，JavaServer Pages，Java Expression Language和Java WebSocket规范是在[Java Community Process]下开发的 。它是一个开源的轻量级Web应用服务器，在中小型系统和并发量小的场合下被普遍使用，是开发和调试Servlet、JSP 程序的首选。

Tomcat主要组件：服务器Server，服务Service，连接器Connector、容器Container。连接器Connector和容器Container是Tomcat的核心。

一个Container容器和一个或多个Connector组合在一起，加上其他一些支持的组件共同组成一个Service服务，有了Service服务便可以对外提供能力了，但是Service服务的生存需要一个环境，这个环境便是Server，Server组件为Service服务的正常使用提供了生存环境，Server组件可以同时管理一个或多个Service服务。

搭建

本项目使用的版本是Tomcat 8.5.50（https://tomcat.apache.org/download-80.cgi）

目录结构如下所示

tomcat

├── BUILDING.txt

├── CONTRIBUTING.md

├── LICENSE

├── NOTICE

├── README.md

├── RELEASE-NOTES

├── RUNNING.txt

├── bin

│   ├── attachments

│   ├── bootstrap.jar

│   ├── catalina-tasks.xml

│   ├── catalina.sh

│   ├── ciphers.sh

│   ├── commons-daemon-native.tar.gz

│   ├── commons-daemon.jar

│   ├── configtest.sh

│   ├── daemon.sh

│   ├── digest.sh

│   ├── hs\_err\_pid69993.log

│   ├── logs

│   │   └── logback

│   │   ├── error-2020-01-06.log

│   │   ├── smile\_2020-1-6.log

│   │   └── spring\_2020-1-6.log

│   ├── logs.log

│   ├── logs.log\_2020-01-19.log\ \

│   ├── logs.log\_2020-02-02.log\ \

│   ├── setclasspath.sh

│   ├── shutdown.sh

│   ├── startup.sh

│   ├── tomcat-juli.jar

│   ├── tomcat-native.tar.gz

│   ├── tool-wrapper.sh

│   └── version.sh

├── conf

│   ├── Catalina

│   │   └── localhost

│   ├── catalina.policy

│   ├── catalina.properties

│   ├── context.xml

│   ├── jaspic-providers.xml

│   ├── jaspic-providers.xsd

│   ├── logging.properties

│   ├── server.xml

│   ├── tomcat-users.xml

│   ├── tomcat-users.xsd

│   └── web.xml

├── lib

├── logs

├── temp

├── webapps

│   ├── ROOT

│   ├── blog

│   ├── blog.war

│   ├── docs

│   ├── examples

│   ├── host-manager

│   ├── manager

└── work

└── Catalina

└── localhost

├── ROOT

├── blog

├── docs

├── examples

├── host-manager

└── manager

```

先把Tomcat的目录的读写权限设置好，方便接下来的设置与部署。

在conf/server.xml中配置端口号，初始堆大小，最大堆大小，以及自动热部署功能，并把本项目设置为Tomcat的默认应用

最后把本项目编译好的blog.war拷贝到wepapps下，启动Tomcat：startup.sh

Tomcat会启动JVM，自动扫描wepapps下的项目并解压，把类加载到堆内存中。

## 2.1 WebSocket简介

​ WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议。

​ WebSocket 使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在 WebSocket API 中，浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。

​ 在 WebSocket API 中，浏览器和服务器只需要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。

​ 现在，很多网站为了实现推送技术，所用的技术都是 Ajax 轮询。轮询是在特定的的时间间隔（如每1秒），由浏览器对服务器发出HTTP请求，然后由服务器返回最新的数据给客户端的浏览器。这种传统的模式带来很明显的缺点，即浏览器需要不断的向服务器发出请求，然而HTTP请求可能包含较长的头部，其中真正有效的数据可能只是很小的一部分，显然这样会浪费很多的带宽等资源。

​ HTML5 定义的 WebSocket 协议，能更好的节省服务器资源和带宽，并且能够更实时地进行通讯。

​ 浏览器通过 JavaScript 向服务器发出建立 WebSocket 连接的请求，连接建立以后，客户端和服务器端就可以通过 TCP 连接直接交换数据。

​ 当你获取 Web Socket 连接后，你可以通过 \*\*send()\*\* 方法来向服务器发送数据，并通过 \*\*onmessage\*\* 事件来接收服务器返回的数据。

​ WebSocket 协议本质上是一个基于 TCP 的协议。

​ 为了建立一个 WebSocket 连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求不同，包含了一些附加头信息，其中附加头信息"Upgrade: WebSocket"表明这是一个申请协议升级的 HTTP 请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket 连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。

## 2.1 JavaWeb原理介绍

Java Web 是用 Java 技术来解决相关 web 互联网领域的技术总和。

Web 包括：web 服务器和 web 客户端两部分。

Java 在客户端的应用有 java applet，使用得很少，Java 在服务器端的应用非常丰富，比如 Servlet，JSP 和第三方框架等等。

1. C/S 体系结构

（1）概念

C/S 是 Client/Server 的缩写，即客户端 / 服务器结构。

（2）特点

这种结构可以充分利用两端硬件环境的优势，将任务合理分配到客户端和服务器，从而降低了系统的通信开销。同时安全性较高。

3. B/S 体系结构

（1）概念

B/S 是 Browser/Server 的缩写，即 浏览器/ 服务器结构。统一采用如 IE、Firefox、Chrome 等浏览器，通过 Web 浏览器向 Web 服务器发送请求，由 Web 服务器进行处理，并将处理结果逐级传回客户端。

（2）特点

​节约了开发成本，是一种全新的软件体系结构。这种体系结构已经成为当今应用软件的首选体系结构。安全性相对C/S较低。

Servlet

​类路径：javax.servlet.HttpServlet

​Servlet是一种独立于平台和协议的服务器端的Java应用程序，可以生成动态的web页面。它担当Web浏览器或其他http客户程序发出请求、与http服务器上的数据库或应用程序之间交互的中间层。

​Servlet是用Java编写的Server端程序，它与协议和平台无关。Servlet运行于Java服务器中。

​Java Servlet可以动态地扩展服务器的能力，并采用请求-响应模式提供Web服务。

​Servlet是使用Java Servlet应用程序设计接口及相关类和方法的Java程序。它在Web服务器上或应用服务器上运行并扩展了该服务器的能力。Servlet装入Web服务器并在Web服务器内执行。

Servlet是以Java技术为基础的服务器端应用程序组件，Servlet的客户端可以提出请求并获得该请求的响应，它可以是任何Java程序、浏览器或任何设备。

​当Web服务器接收到一个HTTP请求时，它会先判断请求内容——如果是静态网页数据，Web服务器将会自行处理，然后产生响应信息；如果牵涉到动态数据，Web服务器会将请求转交给Servlet容器。此时Servlet容器会找到对应的处理该请求的Servlet实例来处理，结果会送回Web服务器，再由Web服务器传回用户端。

​针对同一个Servlet，Servlet容器会在第一次收到http请求时建立一个Servlet实例，然后启动一个线程。第二次收到http请求时，Servlet容器无须建立相同的Servlet实例，而是启动第二个线程来服务客户端请求。所以多线程方式不但可以提高Web应用程序的执行效率，也可以降低Web服务器的系统负担。

JSP（JavaServer Pages）

类路径：javax.servlet.jsp

JSP和Servlet的本质是一样的，因为JSP最终需要编译成Servlet才能运行，换句话说JSP是生成Servler的草稿文件。

​JSP就是在HTML中嵌入Java代码，或者使用JSP标签，包括使用用户自定义标签，从而可以动态的提供内容。早起JSP应用比较广泛，一个web应用可以全部由JSP页面组成，只需要少量的JavaBean即可，但是这样导致了JSP职责过于复杂，这是Java EE标准的出现无疑是雪中送炭，因此JSP慢慢发展成单一的表现技术，不再承担业务逻辑组件以及持久层组件的责任。

原理概述：

JSP的本质是servlet，当用户指定servlet发送请求时，servlet利用输出流动态生成HTML页面。由于包含大量的HTML标签。静态文本等格式导致servlet的开发效率极低，所有的表现逻辑，包括布局、色彩及图像等，都必须耦合在Java代码中，起静态的部分无需Java程序控制，只有那些需要从数据库读取或者需要动态生成的页面内容才使用Java脚本控制。

因此，JSP页面内容有以下两部分组成：

静态部分：HTML标签

动态部分：Java脚本

JSP九大内置对象

out（JspWriter）：　　　　　　　　　　等同与response.getWriter()，用来向客户端发送文本数据；

config（ServletConfig）：　　　　　对应“真身”中的ServletConfig；

page（当前JSP的真身类型）：　　　　当前JSP页面的“this”，即当前对象；

pageContext（PageContext）：　　　页面上下文对象，它是最后一个没讲的域对象；

exception（Throwable）：　　　　　　只有在错误页面中可以使用这个对象；

request（HttpServletRequest）：　　即HttpServletRequest类的对象；

response（HttpServletResponse）：即HttpServletResponse类的对象；

application（ServletContext）：　　即ServletContext类的对象；

session（HttpSession）：　　　　　即HttpSession类的对象，不是每个JSP页面中都可以使用，如果在某个JSP页面中设置<%@page session=”false”%>，说明这个页面不能使用session。

## 2.1 JavaBean简介

JavaBean 是一种JAVA语言写成的可重用组件。为写成JavaBean，类必须是具体的和公共的，并且具有无参数的构造器。JavaBean 通过提供符合一致性设计模式的公共方法将内部域暴露成员属性，set和get方法获取。众所周知，属性名称符合这种模式，其他Java 类可以通过自省机制(反射机制)发现和操作这些JavaBean 的属性。

JavaBean是一种可重用的Java组件，它可以被Applet、Servlet、SP等Java应用程序调用．也可以可视化地被Java开发工具使用。它包含属性(Properties)、方法(Methods)、事件(Events)等特性。

JavaBean是一种软件组件模型，就跟ActiveX控件一样，它们提供已知的功能，可以轻松重用并集成到应用程序中的java类。任何可以用Java代码创造的对象都可以利用JavaBean进行封装。通过合理的组织具有不同功能的JavaBean，可以快速的生成一个全新的应用程序，如果将这个应用程序比作一辆汽车，那么这些JavaBean就好比组成这辆汽车的不同零件。对软件开发人员来说，JavaBean带来的最大的优点是充分提高了代码的可重用性，并且对软件的可维护性和易维护性起到了积极作用。

JavaBean的种类按照功能可以划分为可视化和不可视化两类。可视化的JavaBean就是拥有GUI图形用户界面的，对最终用户是可见的。不可视化的JavaBean不要求继承，它更多的被使用在JSP中，通常情况下用来封装业务逻辑、数据分页逻辑、数据库操作和事物逻辑等，这样可以实现业务逻辑和前台程序的分离，提高了代码的可读性和易维护性，使系统更健壮和灵活。随着JSP的发展，JavaBean更多的应用在非可视化领域，并且在服务器端应用方面表现出了越来越强的生命力。

## 2.1 AJAX简介

AJAX即“Asynchronous JavaScript and XML”（异步的[JavaScript](https://zh.wikipedia.org/wiki/JavaScript)与[XML](https://zh.wikipedia.org/wiki/XML)技术），指的是一套综合了多项技术的[浏览器](https://zh.wikipedia.org/wiki/%E7%80%8F%E8%A6%BD%E5%99%A8)端[网页](https://zh.wikipedia.org/wiki/%E7%B6%B2%E9%A0%81)开发技术。Ajax的概念由[杰西·詹姆士·贾瑞特](https://zh.wikipedia.org/wiki/%E5%82%91%E8%A5%BF%C2%B7%E8%A9%B9%E5%A7%86%E5%A3%AB%C2%B7%E8%B3%88%E7%91%9E%E7%89%B9)所提出。

传统的Web应用允许用户端填写表单（form），当提交表单时就向[网页服务器](https://zh.wikipedia.org/wiki/%E7%B6%B2%E9%A0%81%E4%BC%BA%E6%9C%8D%E5%99%A8)发送一个请求。服务器接收并处理传来的表单，然后送回一个新的网页，但这个做法浪费了许多带宽，因为在前后两个页面中的大部分[HTML](https://zh.wikipedia.org/wiki/HTML)码往往是相同的。由于每次应用的沟通都需要向服务器发送请求，应用的回应时间依赖于服务器的回应时间。这导致了用户界面的回应比本机应用慢得多。

与此不同，AJAX应用可以仅向服务器发送并取回必须的数据，并在客户端采用JavaScript处理来自服务器的回应。因为在服务器和浏览器之间交换的数据大量减少，服务器回应更快了。同时，很多的处理工作可以在发出请求的[客户端](https://zh.wikipedia.org/wiki/%E5%AE%A2%E6%88%B7%E7%AB%AF)机器上完成，因此Web服务器的负荷也减少了。

类似于[DHTML](https://zh.wikipedia.org/wiki/DHTML)或[LAMP](https://zh.wikipedia.org/wiki/LAMP)，AJAX不是指一种单一的技术，而是有机地利用了一系列相关的技术。虽然其名称包含XML，但实际上数据格式可以由[JSON](https://zh.wikipedia.org/wiki/JSON)代替，进一步减少数据量，形成所谓的AJAJ。而客户端与服务器也并不需要异步。一些基于AJAX的“派生／合成”式（derivative/composite）的技术也正在出现，如[AFLAX](https://zh.wikipedia.org/wiki/AFLAX)

# 3需求分析

在系统工程及软件工程中，需求分析指的是在创建一个新的或改变一个现存的系统或产品时，确定新系统的目的、范围、定义和功能时所要做的所有工作。需求分析是软件工程中的一个关键过程。在这个过程中，系统分析员和软件工程师确定顾客的需要。只有在确定了这些需要后他们才能够分析和寻求新系统的解决方法。

在软件工程的历史中，很长时间里人们一直认为需求分析是整个软件工程中最简单的一个步骤，但在过去十年中越来越多的人认识到它是整个过程中最关键的一个过程。假如在需求分析时，分析者们未能正确地认识到顾客的需要的话，那么最后的软件实际上不可能达到顾客的需要，或者软件无法在规定的时间里完工。

## 3.1可行性分析

此微博系统将用于以下几个方面

纪录个人生活点滴

如同日记一般，事实上，博客 (blog) 原本的字义 (weblog) 就是网络日志 (web + log)，可将自己每日生活中发生的事情记录下来，并且跟朋友或网民们分享。

传达个人自由思想

博客是个自由的空间，可依照个人自由意志去撰写任意的内容，传达个人的自由思想，举凡政治、经济、社论、科技等等，在民主自由国度也常出现批评政府施政为主题的博客，不过要注意言论之中是否有牵涉到法律诽谤、妨碍风化、知识产权等问题。

抒发与分享个人情绪

博客常用来抒发个人情绪，把生活中的不满、郁闷或愤怒宣泄到博客上，或是与网络上的网民们分享个人开心的心情。

分享与交流知识、技术

如果有专长、兴趣的人，可以借由撰写博客来分享自己的知识与技术，并和网络上的网民交流讨论。

认识同好、结交朋友[[编辑](https://zh.wikipedia.org/w/index.php?title=%E7%B6%B2%E8%AA%8C&action=edit&section=10)]

借由博客的社群性质可以吸引兴趣或性格相合的网民来观看，并且借此认识结交新朋友。

## 3.2安全性分析

“安全性分析”(safety analysis)是- -种系统性的分析，应在研发过程的早期开始进行，用于确定产品在每一-个 使用模式中执行其功能的方式，识别潜在的危险，预计这些危险对人员及(或)设备可能造成的损害，并确定消除危险的方法。其中对于计算机系统来说，安全性分析的一项重要内容是“软件安全性分析”，这是对软件程序进行的一种分析，以保证程序在其设计的运行环境中，不会引起(或可以容忍的小概率引起)或诱发对人员或设备的危害。例如多级火箭- -级点火、二级点火指令如果错了，火箭就会失败。但只要对火箭指令及传递机构采取足够的防错设计，错发指令的概率就可以小到能容忍的程度。如果各关键项目的开发单位能从软件安全性这方面重视“安全”这个题目，那么项目的安全性链条就不会轻易地由于诸如小数点错位的原因而断开。

在软件和信息系统的开发过程中，由于技术难度高，项目复杂，开发周期短而带来的一系列困难，潜伏安全性隐患的几率其实是很大的。现代化的软件本身变得越来越复杂，开发一-个软件产品或--个大型系统所需要依靠的技术也越来越多样化，需要考虑的问题也越来越多，例如，开发团队需要在研发开始前就确定好软件系统能够承受的出事概率。很多软件开发的组织由于没有掌握和利用必要的控制软件安全性的技术,无法要善解决相应的问题，把时间耗费在事后补救上，使得开发的效率大为降低，产品的质量大打折扣,甚至因为某个关键错误的发生，导致产品的信誉度降低，更严重的结果则会导致生命财产安全的损失。如果你发现有关安全性的要求已经出现在安全相关软件项目的合同书或任务书中，并提出软件安全性分析的范围和任务,那么说明已经需要开始进行软件安全性分析的准备了。

此微博系统将涉及和储存大量用户的个人信息，而且系统流程较为复杂，所以安全性显得尤为重要。尤其是用户名，密码，博客内容等私密信息，所以本系统必须通过强大的权限管理以及授权和认证来保证用户的隐私安全。对敏感信息，比如用户名和密码等进行加密传输，重要信息，比如博客内容等进行备份，防止丢失。

## 3.3性能需求分析

为了保证此微博系统的高性能，是十分重要的，良好且高可用的性能是一个优秀的微博系统不可或缺的条件，试想一个拥有众多用户的社交系统，怎么可能会在使用时让用户感受到明显的卡顿和延迟呢。

对于性能一词，人们往往觉得它过于抽象，而延迟和吞吐量，则是性能的具体表现。

Latency，中文译作延迟。Throughput，中文译作吞吐量。它们是衡量软件系统的最常见的两个指标。

延迟一般包括单向延迟（One-way Latency）和往返延迟（Round Trip Latency），实际测量时一般取往返延迟。它的单位一般是ms、s、min、h等。

吞吐量一般指相当一段时间内测量出来的系统单位时间处理的任务数或事务数（TPS）。注意“相当一段时间”，不是几秒，而可能是十几分钟、半个小时、一天、几周甚至几月。它的单位一般是TPS、每单位时间写入磁盘的字节数等。

此微博系统中为了保证良好的性能，采用了包含但不限于以下方案。

1. 单例：Spring的Bean默认的作用域就是单例，使用该属性定义Bean时，IOC容器仅创建一个Bean实例，无论多少次请求获取该Bean，IOC容器每次返回的是同一个Bean实例。
2. WebSocket + 异步AJAX：WebSocket不依赖于HTTP协议，因此建立连接以后不需要一个Response对应一个Request，而是直接依赖于TCP协议解决客户端发起多个http请求到服务器资源浏览器必须要经过长时间的轮训问题而生的，他实现了多路复用，他是全双工通信。在WebSocket协议下客服端和浏览器可以同时发送信息。建立了WenSocket之后服务器不必在浏览器发送request请求之后才能发送信息到浏览器。这时的服务器已有主动权想什么时候发就可以发送信息到服务器。而且信息当中不必在带有head的部分信息了与http的长链接通信来说，这种方式，不仅能降低服务器的压力。而且信息当中也减少了部分多余的信息。
3. 并发容器：在多人聊天室设计中，为了保证并发性能，采用了JDK java.util.concurrent包下的并发容器ConcurrentHashMap，和保证原子性的int包装类AtomicInteger。ConcurrentHashMap在JDK1.8中其中抛弃了原有的Segment 分段锁，而采用了 CAS + synchronized 来保证并发安全性，有效地提升了性能。而AtomicInteger原子操作类利用了CAS在保证线程安全的同时，大大减少了线程因资源竞争而造成的性能消耗。
4. MyBatis缓存：MyBatis默认开启一级缓存，同一个 SqlSession 对象， 在参数和 SQL 完全一样的情况先，只执行一次 SQL 语句（如果缓存没有过期）在同一个 SqlSession 中, Mybatis 会把执行的方法和参数通过算法生成缓存的键值，将键值和结果存放在一个 Map 中，如果后续的键值一样，则直接从 Map 中获取数据；不同的 SqlSession 之间的缓存是相互隔离的；用一个 SqlSession，可以通过配置使得在查询前清空缓存；任何的 UPDATE, INSERT, DELETE 语句都会清空缓存。

## >>>>>>>>>>>3.4技术可行性

Java语言，因为其独特的优点，已经在越来越的高校都开设了这个课程，同时由十应用的需求，也越来越多的人在不断研究J2EE的相关技术，也因此形成了一些比较优秀的应用框架，其中包括Struts, Spring等，通过这些框架的应用，可以更好的继承优秀的设计模块，更好的为程序的稳定性、健壮性及可扩展性服务，从而可以更好的利用这些优秀模块实现系统应用。除此之外，因为Java语言已经有相当长的历史，不管是学习还是应用，都有许多可借鉴的现成实例可用，因此对系统的实现具有很大的帮助。

## 3.5功能需求分析

微博系统是指一种基于用户关系信息分享、传播以及获取的通过关注机制分享简短实时信息的广播式的社交媒体、网络平台，允许用户通过Web、PC、手机等多种移动终端接入，以文字、图片、视频等多媒体形式，实现信息的即时分享、传播互动。

此微博系统主要包含以下功能

1. 主要功能

发布微博

浏览微博

个人空间

粉丝关注

多人聊天

2. 用户登录

除了通过注册来添加用户，管理员也可以添加用户。登录后的用户可以进入并使用本系统。

3. 修改密码

用户和管理员都可以修改密码。

4. 粉丝关注

用户可以关注自己喜欢的博主，并能够在关注动态看到他们新发的微博。

用户还能在关注列表里看到自己关注的所有博主，也可以随时取消关注。

博主也可以在个人空间看到有谁关注了自己。

5. 用户管理

删除用户: 管理员可以删除普通用户。

禁言用户: 使普通用户（永久或期限）禁止在本系统发布任何内容。

查询用户: 通过用户名或ID查找用户，并查看他们的所有信息。

删除博客: 管理员可以删除任何微博。

管理员除了拥有以下功能以外，同时拥有普通用户的所有功能。

6. 微博系统

发布微博: 用户登录后随时可以发布微博（限制140字，9张图片）。

查看微博: 用户可以在首页看到微博，也可以在关注动态看到关注的博主发的微博。

个人空间: 用户可以进入自己或者别人的个人看见，并查看他的信息。

编辑微博: 用户只能编辑自己已经发出的微博。

删除微博: 普通用户只能删除自己发的微博，而管理员能删除任何微博。

7. 用户评论

用户可以在博客下面发表评论，修改或删除自己发的评论。

8. 多人聊天

用户可以在多人聊天室参与聊天，所有进入聊天室的用户都可以实时看到聊天内容。（聊天室属于公开场合，禁言不会影响用户进入聊天室和参与多人聊天）

9. 站内搜索

用户可以通过关键词索用户或者微博内容。

## >>>>>>>>>>>3.6网络原理图

本系统的网络原理图如图2-1所示

用户

局域网或Internet

同学聚会交友系统

系统服务器

后台数据库

图2-1系统的网络原理图

# 4数据库设计

依据系统功能实现需求，数据库设计如下：

## 4.1用户

属性：

+ 用户ID：user\_id

+ 用户名：user\_name

+ 用户密码：user\_password

+ 禁言截止日期：user\_date

+ 启用状态：status

+ 权限：admin

+ 禁言状态：muted

表设计：

```mysql

CREATE TABLE `user` (

`user\_id` int(11) NOT NULL AUTO\_INCREMENT,

`user\_name` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL,

`user\_password` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL,

`user\_date` datetime DEFAULT NULL COMMENT '禁言截止日期，muted为1时才有效',

`status` int(2) NOT NULL DEFAULT '1' COMMENT '1正常，0停用',

`admin` int(2) NOT NULL DEFAULT '0' COMMENT '管理员权限',

`muted` int(2) NOT NULL DEFAULT '0' COMMENT '1禁言，0正常',

PRIMARY KEY (`user\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=56 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci

```

实体类设计：

```java

/\*\*

\* 用户实体类

\*/

public class User {

/\*\*

\* 用于数据库映射

\*/

private int id;

private int status;//状态 1正常，0停用:无法登录

private int admin;//权限

private int muted;//1禁言，0正常

private Date muteDeadline;//禁言截止时间 muted为1时才有效

/\*\*

\* 用于前端显示

\*/

private String muteDeadTime;//muteDeadline 面向前端

/\*\*

\* 其他

\*/

private int fansCount;

private int idolCount;

public int getFansCount() {

return fansCount;

}

public void setFansCount(int fansCount) {

this.fansCount = fansCount;

}

public int getIdolCount() {

return idolCount;

}

public void setIdolCount(int idolCount) {

this.idolCount = idolCount;

}

public String getMuteDeadTime() {

if (muteDeadTime == null && muteDeadline != null)

muteDeadTime = ToolsUtil.SIMPLE\_DATE\_FORMAT.format(muteDeadline);

return muteDeadTime;

}

public void setMuteDeadTime(String muteDeadTime) {

this.muteDeadTime = muteDeadTime;

}

public Date getMuteDeadline() {

return muteDeadline;

}

public void setMuteDeadline(Date muteDeadline) {

this.muteDeadline = muteDeadline;

}

public int getMuted() {

return muted;

}

public void setMuted(int muted) {

this.muted = muted;

}

@NotNull

@Size(min = 1)

@Pattern(regexp = "[\\S]+")

private String name;//用户名

@NotNull

@Size(min = 1)

@Pattern(regexp = "[\\S]+")

private String password;//密码

public User() { }

public User(int id, String name) {

this.id = id;

this.name = name;

}

public int getAdmin() {

return admin;

}

public void setAdmin(int admin) {

this.admin = admin;

}

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "{" + id + ":" + name + "}";

}

@Override

public boolean equals(Object o) {

if (this == o)

return true;

if (!(o instanceof User))

return false;

User user = (User) o;

return id == user.getId() || Objects.equals(name, user.getName());

}

}

```

实体属性图：

<img src="/Users/arvin/Documents/Documents/学校/毕设/images/用户.png" alt="用户" style="zoom:50%;" />

## 4.1博客

属性：

+ 博客ID：blog\_id

+ 博客内容：blog\_content

+ 发布时间：blog\_time

+ 发布者用户ID：user\_id

+ 图片文件名：filenames

+ 状态：status

表设计：

```mysql

CREATE TABLE `blogs` (

`blog\_id` int(11) NOT NULL AUTO\_INCREMENT,

`blog\_content` varchar(127) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL COMMENT '内容',

`blog\_time` datetime DEFAULT NULL COMMENT '发布时间',

`user\_id` int(11) DEFAULT NULL COMMENT '用户',

`filenames` varchar(510) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci DEFAULT NULL COMMENT '图片的文件名',

`status` int(2) NOT NULL DEFAULT '1' COMMENT '1正常，0不展示',

PRIMARY KEY (`blog\_id`),

KEY `user\_id` (`user\_id`),

CONSTRAINT `blogs\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=224 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci

```

实体类设计：

```java

/\*\*

\* 博客实体类

\*/

public class Blog {

/\*\*

\* 用于数据库映射

\*/

private int id;

private String content;//博客内容

private Date date;//时间，面向数据库

private User user;//此博客发布者

private int status;//状态 1显示 0不显示

private String filenames;//图片文件名，面向数据库

private List<Comment> commentList;

/\*\*

\* 用于前端显示

\*/

private String time;//时间，面向前端

private String[] filenameList;//图片文件名，面向前端

private int commentCount;

public void setFilenames(String filenames) {

this.filenames = filenames;

if (filenames != null)

filenameList = filenames.split(" \n ");

}

public void setDate(Date date) {

this.date = date;

}

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

public String[] getFilenameList() {

return filenameList;

}

public String getTime() {

if (time == null && date != null)

time = ToolsUtil.SIMPLE\_DATE\_FORMAT.format(date);

return time;

}

public void setTime(String time) {

this.time = time;

}

public String getFilenames() {

return filenames;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getContent() {

return content;

}

public void setContent(String content) {

this.content = content;

}

public Date getDate() {

return date;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public List<Comment> getCommentList() {

return commentList;

}

public void setCommentList(List<Comment> commentList) {

this.commentList = commentList;

}

public int getCommentCount() {

return commentList.size();

}

@Override

public String toString() {

return id + ":" + content + ":" + ToolsUtil.SIMPLE\_DATE\_FORMAT.format(date) + "{" + user + "}";

}

}

```

实体属性图：

<img src="/Users/arvin/Documents/Documents/学校/毕设/images/博客.png" alt="博客" style="zoom:50%;" />

## 4.1评论

属性：

+ 评论ID：comment\_id

+ 所属博客ID：blog\_id

+ 评论者用户ID：user\_id

+ 评论内容：comment\_content

+ 发布评论时间：comment\_time

+ 状态：status

表设计：

```mysql

CREATE TABLE `comment` (

`comment\_id` int(11) NOT NULL AUTO\_INCREMENT,

`blog\_id` int(11) NOT NULL,

`user\_id` int(11) NOT NULL,

`comment\_content` varchar(127) NOT NULL,

`comment\_time` datetime DEFAULT NULL,

`status` int(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`comment\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=135 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

```

实体类设计：

```java

/\*\*

\* 评论实体类

\*/

public class Comment

{

/\*\*

\* 用于数据库映射

\*/

private int id;

private String content;//评论内容

private Date date;//时间，面向数据库

private User user;//此评论发布者

private int status;//状态 1显示 0不显示

private Blog blog;//对应博客

/\*\*

\* 用于前端显示

\*/

private String time;//时间，面向前端

private static final SimpleDateFormat SIMPLE\_DATE\_FORMAT = new SimpleDateFormat("YYYY年MM月dd日 HH:mm");

public int getId()

{

return id;

}

public void setId(int id)

{

this.id = id;

}

public String getContent()

{

return content;

}

public void setContent(String content)

{

this.content = content;

}

public Date getDate()

{

return date;

}

public void setDate(Date date)

{

this.date = date;

if (date != null)

time = SIMPLE\_DATE\_FORMAT.format(date);

}

public User getUser()

{

return user;

}

public void setUser(User user)

{

this.user = user;

}

public String getTime()

{

return time;

}

public void setTime(String time)

{

this.time = time;

}

public int getStatus()

{

return status;

}

public void setStatus(int status)

{

this.status = status;

}

public Blog getBlog() {

return blog;

}

public void setBlog(Blog blog) {

this.blog = blog;

}

@Override

public String toString()

{

return "Comment{" +

"id=" + id +

", content='" + content + '\'' +

", user=" + user +

", time='" + time + '\'' +

'}';

}

}

```

实体属性图：

<img src="/Users/arvin/Documents/Documents/学校/毕设/images/评论.png" alt="评论" style="zoom:50%;" />

## 4.1关注

属性：

+ 关注ID：attention\_id

+ 博主用户ID：blogger\_id

+ 粉丝用户ID：fans\_id

+ 状态：status

表设计：

```mysql

CREATE TABLE `attention` (

`attention\_id` int(11) NOT NULL AUTO\_INCREMENT,

`blogger\_id` int(11) NOT NULL,

`fans\_id` int(11) NOT NULL,

`status` int(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`attention\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=45 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

```

实体属性图：

<img src="/Users/arvin/Documents/Documents/学校/毕设/images/关注.png" alt="关注" style="zoom:50%;" />

### 实体关系图

![实体关系图](/Users/arvin/Documents/Documents/学校/毕设/images/实体关系图.png)

# 5项目实现

## 5.1项目结构

1. 设计maven的模块结构和依赖关系

项目分为三个模块：

> \*\*blog-model\*\* 负责数据、实体、模型

>

> \*\*blog-service\*\* 封装可复用的、高可用的业务操作

>

> \*\*blog-web\*\* 负责解析视图，进行页面的渲染（SpringMVC直接与这一层交互）

2. 完成 `SpringMVC`（基于注解）的搭建

本项目基于 `Web 3.0` 抛弃了基于繁琐的 `web.xml` 的形式的旧版，使用更加方便，快捷，现代 `SpringMVC` 类库中提供的现有类来配置 `DispacherServlet`

扩展AbstractAnnotationConfigDispatcherServletInitializer类会自动配置DispatcherServlet和Spring应用上下文

```java

/\*\*

\* 继承AbstractAnnotationConfigDispatcherServletInitializer来初始化WEB

\* 扩展AbstractAnnotationConfigDispatcherServletInitializer的任意类都会自动配置DispatcherServlet和Spring应用上下文

\*/

public class BlogWebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

protected String[] getServletMappings() {

return new String[]{"/"};

}

protected Class<?>[] getRootConfigClasses() {

return new Class<?>[]{RootConfig.class};

}

protected Class<?>[] getServletConfigClasses() {

return new Class<?>[]{WebConfig.class};

}

/\*\*

\* 文件上传的临时路径

\*/

@Override

protected void customizeRegistration(ServletRegistration.Dynamic registration) {

registration.setMultipartConfig(new MultipartConfigElement("/arvin/file/images/"));

}

/\*\*

\* 这项配置使得当handlerMapping（处理器适配）无法找到合适的handler（处理器时）

\* 会抛出异常，方便我们后序的捕获处理

\*/

@Override

protected FrameworkServlet createDispatcherServlet(WebApplicationContext servletAppContext) {

final DispatcherServlet dispatcherServlet = (DispatcherServlet) super.createDispatcherServlet(servletAppContext);

dispatcherServlet.setThrowExceptionIfNoHandlerFound(true);

return dispatcherServlet;

}

}

```

3. 完成三层架构的搭建

+ Controller

1. DispacherServlet收到发送的HttpServletRequest，调用HandlerMapping处理器映射器。

2. 处理器映射器找到具体的处理器(可以根据xml配置、注解进行查找)，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。

3. DispatcherServlet调用HandlerAdapter处理器适配器。

4. HandlerAdapter经过适配调用具体的处理器(Controller，也叫后端控制器)。

5. Controller执行完成返回ModelAndView。

业务模块流程。我经常喜欢用控制视图的跳转来简单形容，但是这个是不全面的，因为他除了控制视图的转换之外，还控制了业务的逻辑，但是，这里的控制业务逻辑不是业务逻辑的实现，而仅仅是一个大的模块，你看到之后，知道它实现了这个业务逻辑，但是怎么实现的，不需要关心，仅仅需要调用service层里的一个方法即可，这样使controller层看起来更加清晰。

controller层负责具体的业务模块流程的控制，在此层要调用service层的接口来控制业务流程，控制的配置也同样是在Spring的配置文件里进行，针对具体的业务流程，会有不同的控制器。我们具体的设计过程可以将流程进行抽象归纳，设计出可以重复利用的子单元流程模块。这样不仅使程序结构变得清晰，也大大减少了代码量。

```

controller

├── admin

│   └── AdminController.java

├── blog

│   ├── BlogsController.java

│   └── CommentController.java

├── error

│   └── ErrorController.java

├── fun

│   ├── PageController.java

│   ├── RredirectController.java

│   └── SearchController.java

├── other

│   └── DownloadController.java

└── user

├── AttentionController.java

├── LoginController.java

└── RegisteController.java

```

+ Service

封装了一系列的可复用的业务操作功能

service层主要负责业务模块的应用逻辑应用设计。同样是首先设计接口，再设计其实现类，接着再Sprin

的配置文件中配置其实现的关联。这样我们就可以在应用中调用service接口来进行业务处理。service层

的业务实，具体要调用已经定义的dao层接口，封装service层业务逻辑有利于通用的业务逻辑的独立性和

重复利用性。程序显得非常简洁。

```

service

├── config

│   ├── GetHttpSessionConfigurator.java

│   ├── RootConfig.java

│   └── WebSocket.java

├── dao

│   ├── BlogDao.java

│   ├── CommentDao.java

│   └── UserDao.java

└── service

├── BlogService.java

├── CommentService.java

├── UserService.java

└── impl

├── BlogServiceImpl.java

├── CommentServiceImpl.java

└── UserServiceImpl.java

```

## 5.1数据库持久化

1. Mysql的安装（https://dev.mysql.com/downloads/mysql/）

本项目安装了MySQL5.7，这个版本比较稳定，而且安全性方面有所增强，同时InnoDB首次支持了全文索引

2. 设置mysql，最大连接时间，编码格式等

配置MySQL的用户名，这里用的root账户，配置好密码，并设置最长连接时间为8小时。并调整时区为北京时间（与操作系统一致），根据机器性能设置最大并发数量，并把所有编码格式都调整为utf-8mb4（Mysql的utf8编码并不是真正的UTF-8编码，Mysql的utf8最多只支持3个字节，而emoji表情、一些特殊的中文字符则需要4个字节才能存储， 因此才会报错。下面是来自[维基百科的Unicode字符平面映射](https://zh.wikipedia.org/wiki/Unicode字符平面映射)，其中UTF-8编码是U+2528D，属于CJK Unified Ideographs Extension B（中日韩统一表意文字扩充B）字符集的字符，处于第二辅助平面（SIP，表意文字补充平面），最多支持4个字节。而Mysql的utf8编码则属于常见的基本多文种平面（BMP，即Unicode编码范围在0000-FFFF之内）的字符，最多支持3个字节。）

3. 设置默认引擎，缓存配置，以及最大并发量

4. 表设计

设计主要表的第一个版本：用户，博客，评论，关注

5. 添加测试数据

手动添加测试数据到数据库

6. 数据库移动到服务器

用MySQL的dump命令导出blog.sql文件，在Linux云主机上导入初始测试数据

7. 构建MyBatis测试

写一个简单的MyBatis测试，测试其久化功能是否正常，性能是否足够

## 5.1日志

1. Log4j介绍

Apache Log4j 2是Log4j的升级版，对Log4j的前身Log4j 1.x进行了重大改进，并提供了Logback中可用的许多改进，同时解决了Logback体系结构中的一些固有问题。

Log4j的API与实现是分开的，从而使应用程序开发人员可以清楚地了解他们可以使用哪些类和方法，同时确保向前的兼容性。这允许Log4j团队以兼容的方式安全地改进实施。

Log4j API是一个日志外观，可以与Log4j实现一起使用，但是也可以在其他日志实现（例如Logback）之前使用。与SLF4J相比，Log4j API具有多个优点：1. Log4j API支持记录[消息](https://logging.apache.org/log4j/2.x/manual/messages.html)而不只是字符串。2. Log4j API支持lambda表达式。3.与SLF4J相比，Log4j API提供了更多的日志记录方法。4.除了SLF4J支持的“参数化日志记录”格式外，Log4j API还支持使用java.text.MessageFormat语法以及printf样式消息的事件。5. Log4j API提供了LogManager.shutdown（）方法。基础的日志记录实现必须实现Terminable接口才能使该方法生效。6.完全支持标记，日志级别和ThreadContext（aka MDC）之类的其他构造。

Log4j 2包含基于LMAX Disruptor库的下一代异步记录器。在多线程方案中，与Log4j 1.x和Logback相比，异步Logger的吞吐量高18倍，延迟降低了几个数量级。有关详细信息，请参见[异步日志记录性能](https://logging.apache.org/log4j/2.x/manual/async.html#Performance)。否则，Log4j 2明显优于Log4j 1.x，Logback和java.util.logging，尤其是在多线程应用程序中。Log4j 2 API将提供最佳性能，而Log4j 2提供对Log4j 1.2，SLF4J，Commons Logging和java.util.logging（JUL）API的支持。

编码为Log4j 2 API的应用程序始终可以选择使用任何SLF4J兼容库作为其Log4j-to-slf4j适配器的记录器实现。与Logback一样，Log4j 2可以在修改后自动重新加载其配置。与Logback不同，它在进行重新配置时不会丢失日志事件。Log4j 2支持基于上下文数据，标记，正则表达式和Log事件中的其他组件进行过滤。可以指定过滤，以将其应用于所有事件，然后再传递给Logger或当事件通过Appender时。此外，过滤器也可以与Loggers关联。与Logback不同，您可以在任何这些情况下使用通用的Filter类。

在Log4j 2中，可以通过代码或配置轻松定义[自定义日志级别](https://logging.apache.org/log4j/2.x/manual/customloglevels.html)。无需子类化。除了使用Log4j API中的许多日志方法之一之外，还可以使用构建器来构造日志事件。有关更多信息，请参见[Log Builder] [manual / logbuilder.html]。

在稳态日志记录期间，Log4j 2 在独立应用程序中是无[垃圾的](https://logging.apache.org/log4j/2.x/manual/garbagefree.html)，而在Web应用程序中是低垃圾的。这样可以减少垃圾收集器上的压力，并可以提供更好的响应时间性能。

2. 用log4j建立项目日志体系

在类路径下创建log4j.properties，并从官网（https://logging.apache.org/log4j/2.x/）拷贝一份log4j.properties的模板，根据实际要求修改loggers, appenders 和layouts，然后再测试日志在控制台和log文件中是否能够正常输出，磨刀不误砍柴工，日志的配置将大大提升接下来的开发效率。

+ Logger: 日志记录器，日志记录的核心类，用于输出不同日志级别的消息

+ Appender: 日志输出目标，用于指定日志输出的目的地，如控制台、文件等等。

+ Layout: 日志格式化器，用于指定日志按照什么格式输出，是日志输出的格式化器。

# 5系统实现

## 5.1系统总体架构及实现

## 5.1用户登录

输入网址后自动进入登录页面，未登录用户尝试符合/user/\*\*的请求是，会被UserFilter拦截，重定向到登录页面，做到登录校验拦截器的功能。

```java

/\*\*

\* 未登录拦截

\*/

@WebFilter(filterName = "userFilter",urlPatterns = "/user/\*")

public class UserFilter implements Filter

{

public void destroy(){}

public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws ServletException, IOException

{

if (((HttpServletRequest) req).getSession().getAttribute("user") != null)

chain.doFilter(req, resp);

else

((HttpServletResponse)resp).sendRedirect("/");

}

public void init(FilterConfig config){ }

}

```

用户请求登录时，后台验证账户和密码参数，它们都不为空，且验证正确是才回成功登录，清除上一位用户的搜索记录等残留信息，并在会话（HttpSession）中添加登录的用户信息，以便接下来使用和防止被登录拦截器拦截，如果登录失败，则重定向到登录页面，传递错误信息。

```java

/\*\*

\* 登录功能

\*/

@PostMapping({"/", "login.html"})

public String login(@Valid User user, Errors errors, HttpSession session, RedirectAttributes model) {

/\*检查用户名和密码是否为空\*/

if (errors.hasErrors())

return "login";

/\*检查用户名和密码是否正确\*/

User realUser = userService.checkUser(user);

if (realUser != null) {

session.setAttribute("user", realUser);

/\*清除搜索的记录\*/

session.removeAttribute("blogList");

session.removeAttribute("keyword");

session.removeAttribute("page");

return "redirect:user/blogs.html";

}

/\*通知是否有错误\*/

model.addFlashAttribute("msg", "err");

return "redirect:/";

}

```

相关SQL语句

```mysql

SELECT user\_id, user\_name, status, admin, user\_date, muted, user\_password

FROM user

WHERE user\_name = '施航程' AND user\_password = '123' and status > 0;

```

## 5.1修改密码

用户在已经登录的状态下可以在个人空间修改自己的密码，修改之前会校验密码是否符合规范（任意长度的非空字符串），不符合会抛出异常，警告用户。

```java

/\*\*

\* 修改密码

\*/

@PostMapping("user/changePassword")

public String changePassword(HttpSession session, String password) {

if (!Pattern.matches("[\\S]+", password))

throw new IllegalArgumentException("密码格式错误!");

User user = (User) session.getAttribute("user");

user.setPassword(password);

userService.changePassword(user);

return "redirect:/user/page.html";

}

```

相关SQL语句

```mysql

UPDATE user SET user\_password = '123456'

WHERE user\_id = 2;

```

## 5.1粉丝关注

- 用户可以关注自己喜欢的博主，并能够在关注动态看到他们新发的微博。

- 用户还能在关注列表里看到自己关注的所有博主，也可以随时取消关注。

- 博主也可以在个人空间看到有谁关注了自己。

```java

/\*\*

\* 用户关注相关

\*/

@Controller

public class AttentionController {

@Resource

private UserService userService;

/\*\*

\* 关注

\*/

@PostMapping("user/addAttention/{userId}/{page}")

public String addAttention(HttpSession session, @PathVariable int userId, @PathVariable String page) {

User me = (User) session.getAttribute("user");

int myId = me.getId();

if (myId != userId && !userService.isAttention(me.getId(), userId))

userService.addAttention(myId, userId);

return "redirect:/user/page.html/" + userId + "/" + page;

}

/\*\*

\* 取关

\*/

@PostMapping("user/delAttention/{userId}/{page}")

public String delAttention(HttpSession session, @PathVariable int userId, @PathVariable String page) {

User me = (User) session.getAttribute("user");

userService.delAttention(me.getId(), userId);

return "redirect:/user/page.html/" + userId + "/" + page;

}

/\*\*

\* 关注列表

\*/

@GetMapping("user/attentionList/{userId}")

public String attentionList(@PathVariable int userId, Model model) {

PageInfo<User> listPage = userService.attentionList(userId);

model.addAttribute("listPage", listPage);

model.addAttribute("host", userService.findUserById(userId));

model.addAttribute("msg", "的关注");

return "list";

}

/\*\*

\* 粉丝列表

\*/

@GetMapping("user/fansList/{userId}")

public String fansList(@PathVariable int userId, Model model) {

PageInfo<User> listPage = userService.fansList(userId);

model.addAttribute("listPage", listPage);

model.addAttribute("host", userService.findUserById(userId));

model.addAttribute("msg", "的粉丝");

return "list";

}

}

```

相关SQL语句

```xml

<insert id="attention">

INSERT INTO attention(blogger\_id, fans\_id)

VALUES (#{blogger}, #{fans})

</insert>

<delete id="delAttention">

DELETE

FROM attention

WHERE blogger\_id = #{blogger}

AND fans\_id = #{fans}

</delete>

<select id="attentionList" resultMap="userMap" parameterType="int">

select u.user\_id, u.user\_name, u.status, u.admin, u.user\_date, u.muted

from user u

inner join attention a on u.user\_id = a.blogger\_id

where a.fans\_id = #{id}

</select>

<select id="fansList" resultMap="userMap">

select u.user\_id, u.user\_name, u.status, u.admin, u.user\_date, u.muted

from user u

inner join attention a on u.user\_id = a.fans\_id

where a.blogger\_id = #{id}

</select>

```

## 5.1用户管理

- 删除用户: 管理员可以删除普通用户。

- 禁言用户: 使普通用户（永久或期限）禁止在本系统发布任何内容。

- 查询用户: 通过用户名或ID查找用户，并查看他们的所有信息。

- 删除博客: 管理员可以删除任何微博。

- 管理员除了拥有以下功能以外，同时拥有普通用户的所有功能。

非管理员用户尝试符合/admin/\*\*的请求是，会被AdminFilter拦截，重定向到主页面，做到权限校验拦截器的功能。

```java

/\*\*

\* 未登录拦截

\*/

@WebFilter(filterName = "adminFilter",urlPatterns = "/admin/\*")

public class AdminFilter implements Filter

{

public void destroy(){}

public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws ServletException, IOException

{

User user = (User) ((HttpServletRequest) req).getSession().getAttribute("user");

if (user != null && user.getAdmin() > 0)

chain.doFilter(req, resp);

else

((HttpServletResponse)resp).sendRedirect("/user/blogs.html");

}

public void init(FilterConfig config){ }

}

```

用户请求登录时，后台会在会话（HttpSession）中添加登录的用户信息时，加入用户的管理员权限信息，以便接下来使用和被登录拦截器验证通过或者拦截，如果验证失败，则重定向到主页面。

```java

/\*\*

\* 管理员相关

\*/

@Controller

public class AdminController {

@Resource

private UserService userService;

/\*\*

\* 管理员页面,默认第一页

\*/

@GetMapping("admin/manager.html")

public String admin(Model model, @RequestParam(required = false) String username) {

return admin(model, 1, username);

}

/\*\*

\* 管理员页面,分页

\*/

@GetMapping("admin/manager.html/{page}")

public String admin(Model model, @PathVariable int page, @RequestParam(required = false) String username) {

/\*根据页号获取数据\*/

username = StringUtils.isEmpty(username) ? null : username.trim();

PageInfo<User> userPage = userService.selectAllUser(page, username);

model.addAttribute("userPage", userPage);

return "admin/list";

}

/\*\*

\* 删除用户

\*

\* @param userId userId

\* @param page page

\*/

@PostMapping("admin/delUser/{userId}/{page}")

public String delUser(@PathVariable int userId, @PathVariable String page) {

userService.delUser(userId);

return "redirect:/admin/manager.html/" + page;

}

/\*\*

\* 恢复用户

\*

\* @param userId userId

\* @param pages page

\*/

@PostMapping("admin/reuseUser/{userId}/{pages}")

public String reuseUser(@PathVariable int userId, @PathVariable String pages) {

userService.reuseUser(userId);

return "redirect:/admin/manager.html/" + pages;

}

/\*\*

\* 禁言用户

\*

\* @param userId userId

\* @param days 天数

\* @param pages pages

\*/

@PostMapping("admin/mute/{userId}/{days}/{pages}")

public String mute(@PathVariable int userId, @PathVariable int days, @PathVariable String pages) {

userService.mute(userId, days);

return "redirect:/admin/manager.html/" + pages;

}

}

```

相关SQL语句

```xml

<select id="selectAllUser" resultMap="userMap" parameterType="string">

SELECT user\_id, user\_name, status, admin, user\_password, user\_date, muted

FROM user

<where>

<if test="\_parameter != null">

user\_id = #{key}

OR user\_name LIKE &quot;%&quot;#{key}&quot;%&quot;

</if>

</where>

ORDER BY user\_id

</select>

<update id="delUser" parameterType="int">

UPDATE user

SET status = 0

WHERE user\_id = #{id}

</update>

<update id="reuseUser" parameterType="int">

UPDATE user

SET status = 1

WHERE user\_id = #{id}

</update>

<update id="mute" parameterType="com.arvinclub.model.entity.User">

UPDATE user

SET user\_date = #{muteDeadline},

muted = #{muted}

WHERE user\_id = #{id};

</update>

```

## 5.1微博系统

- 发布微博: 用户登录后随时可以发布微博（限制140字，9张图片）。

1. 从会话（Http Session）中获取当前用户信息，包括用户ID、用户名。

2. 为传过来的博客实体添加图片文件信息（如果用户发了图片的话）。

3. MyBatis自动生成的SQL语句，并执行。

4. 重定向到博客浏览页面，用户就能看到自己刚刚发布的博客了。

```java

/\*\*

\* 发布博客

\*

\* @param blog 博客

\* @param files 图片文件

\* @return 成功则转到社区主页，否则转到错误提示页面

\*/

@PostMapping("user/addBlog")

public String addBlog(Blog blog, HttpSession httpSession, @RequestParam MultipartFile[] files) throws Exception {

User me = (User) httpSession.getAttribute("user");

userService.checkMuted(me);

/\*给blog添加filenames（面向数据库的图片文件名）\*/

blog.setFilenames(ToolsUtil.saveIMG(files, CLASSPATH));

blog.setUser(me);

/\*添加博客\*/

blogService.addBlog(blog);

return "redirect:/user/blogs.html";

}

```

相关SQL语句

```xml

<insert id="addBlog">

INSERT INTO blogs (blog\_content, blog\_time, user\_id, filenames)

VALUES (#{content}, #{time}, #{id}, #{filenames})

</insert>

```

- 查看微博: 用户可以在首页看到微博，也可以在关注动态看到关注的博主发的微博。通过页号，查询出一部分信息，并返回，如果未指定页号则默认第一页，

```java

/\*\*

\* 查看博客（默认第一页）

\*/

@GetMapping("user/blogs.html")

public String showBlogs(Model model) {

return showBlogs(model, 1);

}

/\*\*

\* 查看博客（指定页号）

\*

\* @param page 页号

\*/

@GetMapping("user/blogs.html/{page}")

public String showBlogs(Model model, @PathVariable int page) {

/\*根据页号获取数据\*/

PageInfo<Blog> blogPage = blogService.selectAllBlog(page);

model.addAttribute("blogPage", blogPage);

return "blogs";

}

/\*\*

\* 查看关注动态（默认第一页）

\*/

@GetMapping("user/attention.html")

public String attention(Model model,HttpSession session) {

return attention(model, 1, session);

}

/\*\*

\* 查看关注动态（指定页号）

\*

\* @param page 页号

\*/

@GetMapping("user/attention.html/{page}")

public String attention(Model model, @PathVariable int page,HttpSession session) {

/\*根据页号获取数据\*/

User me = (User) session.getAttribute("user");

PageInfo<Blog> blogPage = blogService.attentionBlogs(me.getId(), page);

model.addAttribute("blogPage", blogPage);

return "attention";

}

```

相关SQL语句

```xml

<select id="selectBlogsByUserId" resultMap="blogMap" parameterType="int">

SELECT blog\_id, blog\_content, blog\_time, user\_id, filenames

FROM blogs

<where>

<if test="\_parameter > 0">

user\_id = #{id}

</if>

AND status = 1

</where>

ORDER BY blog\_id DESC

</select>

<select id="attentionBlogs" resultMap="blogMap" parameterType="int">

SELECT blog\_id, blog\_content, blog\_time, user\_id, filenames

FROM blogs

WHERE status = 1

AND user\_id IN (SELECT blogger\_id FROM attention WHERE fans\_id = #{id} and status = 1)

ORDER BY blog\_id DESC

</select>

```

+ 个人主页: 用户可以查看其他用户的个人主页，会显示他的所有博客。通过参数中的用户ID，查找某个用户的博客，如果该用户不存在，则抛出异常。通过页号，查询出一部分信息，并返回，如果未指定页号则默认第一页。

```java

/\*\*

\* 个人主页相关

\*/

@Controller

public class PageController {

@Resource

private BlogService blogService;

@Resource

private UserService userService;

/\*\*

\* 进入我的个人主页（默认第一页）

\*/

@GetMapping("user/page.html")

public String seeMe(Model model, HttpSession session) {

return seeWithPage(0, model, "1", session);

}

/\*\*

\* 进入某人个人主页（默认第一页）

\*/

@GetMapping("user/page.html/{userID}")

public String see(@PathVariable int userID, Model model, HttpSession session) {

return seeWithPage(userID, model, "1", session);

}

/\*\*

\* 进入某人个人主页（指定页号）

\*/

@GetMapping("user/page.html/{userID}/{str}")

public String seeWithPage(@PathVariable int userID, Model model, @PathVariable String str, HttpSession session) {

/\* 确认用户\*/

User me = (User) session.getAttribute("user");

User someone = userService.findUserById(userID == 0 ? me.getId() : userID);

/\*确认页号,开始查询\*/

PageInfo<Blog> blogPage = blogService.selectBlogsByUserId(someone.getId(), Integer.parseInt(str));

/\*返回信息\*/

model.addAttribute("blogPage", blogPage);

model.addAttribute("someone", someone);

boolean hoster = me.equals(someone);

model.addAttribute("hoster", hoster);

if (!hoster)

model.addAttribute("isAttention", userService.isAttention(me.getId(), someone.getId()));

return "page";

}

}

```

相关SQL语句

```xml

<select id="selectBlogsByUserId" resultMap="blogMap" parameterType="int">

SELECT blog\_id, blog\_content, blog\_time, user\_id, filenames

FROM blogs

<where>

<if test="\_parameter > 0">

user\_id = #{id}

</if>

AND status = 1

</where>

ORDER BY blog\_id DESC

</select>

```

```mysql

SELECT blog\_id, blog\_content, blog\_time, user\_id, filenames

FROM blogs

WHERE user\_id = 2 AND status = 1 ORDER BY blog\_id DESC

LIMIT 15;

```

- 编辑微博: 用户只能编辑自己已经发出的微博。

1. 获取当前登录的用户。

2. 通过传过来的参数：博客ID确定尝试修改的博客。

3. 获取该博客发布者的用户。

4. 确定当前登录的用户是否有权限更改此博客（当前登录的用户就是发布者，或者权限高于发布者则被判断为有权限）。

5. 无论是否修改成功，都重定向到此博客的详情页面。

```java

/\*\*

\* 编辑博客

\*

\* @return 成功则转到个人主页，否则转到错误提示页面

\*/

@PostMapping("user/editBlog")

public String editBlog(Blog blog, HttpSession httpSession) {

/\*如果有编辑权限则删除\*/

User me = (User) httpSession.getAttribute("user");

userService.checkMuted(me);

Blog oldBlog = blogService.blogDetail(blog.getId());

User someone = oldBlog.getUser();

if (me.equals(someone) || me.getAdmin() > someone.getAdmin()) {

blog.setContent(blog.getContent().replaceAll("\\\\r\\\\n","\r\n"));

blogService.editBlog(blog);

}

return "redirect:/user/detail.html/" + blog.getId();

}

```

相关SQL语句

```xml

<update id="editBlog" parameterType="com.arvinclub.model.entity.Blog">

update blogs

set blog\_content = #{content}

where blog\_id = #{id}

</update>

```

```mysql

update blogs set blog\_content = '测试'

WHERE blog\_id = 223;

```

- 删除微博: 普通用户只能删除自己发的微博，而管理员能删除任何微博。

1. 获取当前登录的用户。

2. 通过传过来的参数：博客ID确定尝试的博客。

3. 获取该博客发布者的用户。

4. 确定当前登录的用户是否有权限删除此博客（当前登录的用户就是发布者，或者权限高于发布者则被判断为有权限）。

5. 无论是否删除成功，都重定向到此博客的详情页面。

```java

/\*\*

\* 删除博客

\*

\* @param blogId blog\_id

\* @return 成功则转到个人主页，否则转到错误提示页面

\*/

@GetMapping("user/delBlog/{blogId}")

public String delBlog(@PathVariable int blogId, HttpSession httpSession) {

/\*如果有编辑权限则删除\*/

Blog blog = blogService.blogDetail(blogId);

User me = (User) httpSession.getAttribute("user");

User someone = blog.getUser();

if (me.equals(someone) || me.getAdmin() > someone.getAdmin())

blogService.delBlog(blogId);

return "redirect:/user/page.html/" + blog.getUser().getId();

}

```

相关SQL语句

删除博客实际上是改变此博客的标志，而非真正删除所有数据

```xml

<update id="delBlog" parameterType="int">

UPDATE blogs

SET status = 0

WHERE blog\_id = #{id}

</update>

```

```mysql

UPDATE blogs SET status = 0

WHERE blog\_id = 223;

```

## 5.1评论系统

- 用户可以在博客下面发表评论，修改或删除自己发的评论。

```java

/\*\*

\* 评论相关

\*/

@Controller

public class CommentController {

@Resource

private CommentService commentService;

@Resource

private UserService userService;

/\*\*

\* 发布评论

\*

\* @return 成功则转到微博详情，否则转到错误提示页面

\*/

@PostMapping("user/addComment/{blogId}")

public String addComment(Comment comment, HttpSession httpSession, @PathVariable int blogId) {

if (StringUtils.isNotBlank(comment.getContent())) {

/\*构建评论\*/

User me = (User) httpSession.getAttribute("user");

userService.checkMuted(me);

comment.setUser(me);

comment.setTime(ToolsUtil.getNowTimeString());

Blog blog = new Blog();

blog.setId(blogId);

comment.setBlog(blog);

commentService.addComment(comment);

}

/\*添加评论\*/

return "redirect:/user/detail.html/" + blogId;

}

/\*\*

\* 删除评论

\*

\* @return 成功则转到微博详情，否则转到错误提示页面

\*/

@GetMapping("user/delComment/{commentId}")

public String delComment(HttpSession httpSession, @PathVariable int commentId) {

Comment comment = commentService.backCommentById(commentId);

User me = (User) httpSession.getAttribute("user");

User someone = comment.getUser();

User blogger = comment.getBlog().getUser();

if (me.equals(someone) || me.getAdmin() > someone.getAdmin() ||

me.equals(blogger) || me.getAdmin() > blogger.getAdmin()) {

commentService.delComment(commentId);

}

return "redirect:/user/detail.html/" + comment.getBlog().getId();

}

/\*\*

\* 编辑评论

\*

\* @return 成功则转到微博详情，否则转到错误提示页面

\*/

@PostMapping("user/editComment")

public String editComment(Comment comment, HttpSession httpSession) {

Comment oldComment = commentService.backCommentById(comment.getId());

User me = (User) httpSession.getAttribute("user");

userService.checkMuted(me);

if (StringUtils.isNotEmpty(comment.getContent()) && me.equals(oldComment.getUser())) {

commentService.updateComment(comment);

}

return "redirect:/user/detail.html/" + oldComment.getBlog().getId();

}

}

```

相关SQL语句

```xml

<update id="updateComment" parameterType="com.arvinclub.model.entity.Comment" useGeneratedKeys="true" keyProperty="id">

update comment

set comment\_content = #{content}

where comment\_id = #{id}

</update>

<delete id="delComment" parameterType="int">

DELETE FROM comment

WHERE comment\_id = #{id}

</delete>

<select id="backCommentById" resultMap="commentMap" parameterType="int">

SELECT comment\_id, comment\_content, comment\_time, user\_id, status, blog\_id

FROM comment

WHERE comment\_id = #{id}

AND status = 1

ORDER BY comment\_id

</select>

<select id="backCommentByBlogId" resultMap="commentMap" parameterType="int">

SELECT comment\_id, comment\_content, comment\_time, user\_id, status, blog\_id

FROM comment

WHERE blog\_id = #{id}

AND status = 1

ORDER BY comment\_id

</select>

<insert id="addComment">

INSERT INTO comment(blog\_id, user\_id, comment\_content, comment\_time)

VALUES (#{blog}, #{user}, #{content}, #{time})

</insert>

<resultMap id="commentMap" type="com.arvinclub.model.entity.Comment">

<id column="comment\_id" property="id" jdbcType="INTEGER"/>

<result column="comment\_content" property="content" jdbcType="VARCHAR" javaType="string"/>

<result column="comment\_time" property="date" jdbcType="TIMESTAMP" javaType="java.util.Date"/>

<result column="status" property="status" jdbcType="INTEGER" javaType="int"/>

<association property="user" select="com.arvinclub.service.dao.UserDao.backUserById" column="user\_id"/>

<association property="blog" select="com.arvinclub.service.dao.BlogDao.backBlogsById" column="blog\_id"/>

</resultMap>

```

### 8. 多人聊天

- 用户可以在多人聊天室参与聊天，所有进入聊天室的用户都可以实时看到聊天内容。

利用了java.util.concurrent包下的并发容器ConcurrentHashMap，和保证原子性的int包装类AtomicInteger，每有一个用户连接，则在在线用户集合：clients:ConcurrentHashMap<String, WebSocket>中添加一个用户信息，包含了用户名与会话。同时同步更新在线人数：onlineCount:AtomicInteger

当用户发送消息时，服务端会接收到消息，并转发给当前登录的所有用户，期间对每个用户单独加锁，既保证了每个用户的会话的同步，又保证了并发性能

用户断开连接时，在线用户集合：clients:ConcurrentHashMap<String, WebSocket>和在线人数：onlineCount:AtomicInteger也会同步更新

为了保证系统的高可用

```java

/\*\*

\* 基于WebSocket的多人聊天室

\*/

@ServerEndpoint(value = "/webSocketByTomcat", configurator = GetHttpSessionConfigurator.class)

public class WebSocket {

/\*\*

\* 聊天室人数

\*/

private static final AtomicInteger onlineCount = new AtomicInteger(0);

/\*\*

\* 当前聊天室成员

\*/

private static final Map<String, WebSocket> clients = new ConcurrentHashMap<>();

/\*\*

\* 用户信息

\*/

private Session session;

private String username;

private final Object lock = new Object();

/\*\*

\* 新建连接

\*/

@OnOpen

public void onOpen(Session session, EndpointConfig config) throws Exception {

//获取HttpSession，并得到用户名

HttpSession httpSession = (HttpSession) config.getUserProperties().get(HttpSession.class.getName());

username = ((User) httpSession.getAttribute("user")).getName();

//登录冲突提醒

if (clients.containsKey(username)) {

//language=HTML

clients.get(username).sendMessageToMe("<span style=\"color:blue\">你已在其他位置登陆</span>");

clients.get(username).session.close();

}

//该用户进入聊天室

this.session = session;

clients.put(username, this);

//聊天室人数调整，并通知其他用户

onlineCount.set(clients.size());

//language=HTML

sendMessageAll("<span style=\"color:green\">[" + username + "] 进入聊天室, 当前人数: " + getOnlineCount() + "</span>");

}

/\*\*

\* 关闭连接

\*/

@OnClose

public void onClose() throws Exception {

clients.remove(username);

onlineCount.set(clients.size());

//language=HTML

sendMessageAll("<span style=\"color:red\">[" + username + "] 离开聊天室, 当前人数: " + getOnlineCount() + "</span>");

if (session != null && session.isOpen()) {

session.close();

}

}

/\*\*

\* 处理收到的消息

\*/

@OnMessage

public void onMessage(String message) throws IOException {

if (message.equalsIgnoreCase("clear!!!"))

for (WebSocket webSocket : clients.values())

webSocket.session.close();

else if (message.equalsIgnoreCase("num!!!")) {

//language=HTML

sendMessageToMe("<span style=\"color:blue\">当前人数: " + getOnlineCount() + "</span>");

for (String u : clients.keySet())

//language=HTML

sendMessageToMe("<span style=\"color:blue\">[" + u + "] </span>");

} else

sendMessageAll("[" + username + "]: " + message);

}

/\*\*

\* 出错也要保证正常关闭和注销

\*/

@OnError

public void onError(Throwable error) throws Exception {

if (session != null && session.isOpen()) {

session.close();

}

}

/\*\*

\* 广播信息

\*/

private void sendMessageAll(String message) throws IOException {

for (WebSocket item : clients.values())

synchronized (item.lock) {

item.session.getBasicRemote().sendText(message);

}

}

/\*\*

\* 打印出提醒（单人）

\*/

private void sendMessageToMe(String message) throws IOException {

synchronized (lock) {

session.getBasicRemote().sendText(message);

}

}

/\*\*

\* 获取当前人数

\*/

public static int getOnlineCount() {

for (String str : clients.keySet())

if (!clients.get(str).session.isOpen())

clients.remove(str);

return onlineCount.get();

}

}

```

为了保证服务端可以拿到用户名，需要在握手时获取HTTP会话：HttpSession（不是同一个会话），并把HTTP会话保存到WebSocket上下文的用户配置参数 UserProperties中，以便后续获取。

```java

/\*\*

\* 用于WebSocket连接时获取HttpSession

\*/

public class GetHttpSessionConfigurator extends Configurator {

/\*\*

\* 获取HttpSession并放到UserProperties中

\*/

@Override

public void modifyHandshake(ServerEndpointConfig sec, HandshakeRequest request, HandshakeResponse response) {

HttpSession httpSession = (HttpSession) request.getHttpSession();

sec.getUserProperties().put(HttpSession.class.getName(), httpSession);

}

}

```

### 9. 站内搜索

+ 用户可以通过关键词索用户和微博内容。

用HTTP会话保存用户的搜索记录，以便用户下次再进入搜索页面时，可以重现上一次搜索的结果

```java

/\*\*

\* 查找内容

\*/

@Controller

public class SearchController {

@Resource

private BlogService blogService;

/\*\*

\* 进入查找内容页面

\*/

@GetMapping("user/search.html")

public String search() {

return "search";

}

/\*\*

\* 按关键词查找内容（默认第一页）

\*/

@PostMapping("user/search.html")

public String search(@RequestParam String keyword, HttpSession session) {

session.setAttribute("keyword", keyword);

return searchByPage(1, session);

}

/\*\*

\* 按关键词查找内容（指定页号）

\*/

@GetMapping("user/search.html/{page}")

public String searchByPage(@PathVariable int page, HttpSession session) {

/\*确认关键词\*/

String keyword = (String) session.getAttribute("keyword");

if (StringUtils.isBlank(keyword))

return "redirect:/user/search.html";

/\*开始查询,返回信息\*/

session.setAttribute("blogPage", blogService.selectBlogsByKey(keyword, page));

return "search";

}

}

```

相关SQL语句

在SQL查询语句的拼接上，占位符使用了#{}而不是${}，#{}使用了预编译SQL语句的形式，不光有效地提升了SQL语句在重复查询上的性能，还能防止被SQL注入。

```xml

<select id="selectBlogsByKey" resultMap="blogMap" parameterType="string">

SELECT blog\_id, blog\_content, blog\_time, user\_id, filenames

FROM blogs a INNER JOIN user b USING(user\_id)

<where>

a.status = 1

AND blog\_content LIKE &quot;%&quot;#{key}&quot;%&quot;

OR user\_name LIKE &quot;%&quot;#{key}&quot;%&quot;

</where>

ORDER BY blog\_id DESC

</select>

```

```mysql

SELECT blog\_id, blog\_content, blog\_time, user\_id, filenames

FROM blogs a

INNER JOIN user b USING(user\_id)

WHERE a.status = 1 AND blog\_content LIKE "%"'test'"%" OR user\_name LIKE "%"'test'"%" ORDER BY blog\_id DESC

LIMIT 15;

```

# >>>>>>>>6系统测试

# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

## >>>>>>>>6.1测试的目的

软件测试从20世纪70年代的首次提出，到2002年被Rick和stefan在《系统的软件测试》中重新定义，越来越被人们所重视。正是当今对软件质量需求的增加，才使得软件测试崛起并快速发展。软件测试的目的就是保证和提高软件产品的质量。

## >>>>>>>>>>6.2软件测试的原则

当今社会计算机软件在医疗，军事，金融等各个方面的广泛应用和深入。使得计算机软件的质量问题成为了我们关注的焦点。

软件测试是对计算机软件进行工程设计，实施与维护计算机软件的整个生命周期，来度量和提高软件的质量的一个过程。它是保证软件产品质量的重要手段之一，它应贯穿于整个软件开发过程，是我们软件工程的重要的组成部分。

软件质量一直是一个模糊的概念。我们又该如何评价一个软件的质量呢。一般我们评价一个软件的好坏都用功能齐全，性能良好，结构合理之类的词语。这些含糊不清的表达无法给软件质量一个科学的评价。我们可以从以下几方面来描述软件质量：

首先看软件是否合乎它的设计需求，一个软件连它的设计需求都不符合，那么它就不具备质量了。所以第一条就是，设计的软件在性能，功能等方面必须满足需求，能够顺利可靠的运行。

评价软件质量的第二条是软件的结构。设计的软件是否方便阅读和理解，是否方便开发人员对它进行维护与修改。软件行业向来人流动大且频繁，所以这一条是软件公司都期望的。

为了增加用户的体验性，软件系统用户界面要易于使用且美观，这是一个软件应该具有的性能。

为了对软件更好的进行配置，管理与更新，使用开发模型进行软件的开发应该是我们首要的选择，并且对于软件的每个生命周期中的各个阶段都要进行规范化和存档。

最后就是为了能够更好的对软件进行测试，提高测试的质量。我们需要遵守一些软件测试的原则。

原则一：软件的缺点是测试出来的。原则二：我们不可能穷尽测试。原则三：测试越早进行越好。原则四：软件的缺陷不是平均分布的，具有集群效应。原则五：测试的活动依赖于测试的内容。原则六：软件测试完美不代表系统可用。原则七：测试要遵守需求。原则八：测试要贯穿于软件的整个生命周期。遵守测试的原则，我们才能能加充分的暴露软件存在的问题和缺陷，并且尽快解决问题，减少缺陷。

## >>>>>>>>>6.3 测试方法

### >>>>>>>>>>>6.3.1按钮，文本框等控件测试

键盘输入字母或者数字。

键盘输入已经存在的文件名。

键盘输入超长字符。例如：在输入框当中输入超出允许字符数的字符，来检查系统是否可以正确的进行处理。

键盘默认值，空白，空格。

尝试不同类型，应该输入字符的地方，输入数字。

合理利用复制粘贴等操作，强行输入系统内不允许的数据。

键盘输入特殊字符。

键盘输入超过文本框长度的文本或字符。

键盘输入不符合格式要求的数据，来检查程序是否可以正常校验。例如：程序要求输入的程序是1995-09-01，实际输入的是95-09-01。程序应该给出错误提示。

### >>>>>>>>>>>>>6.3.2命令按钮控件测试

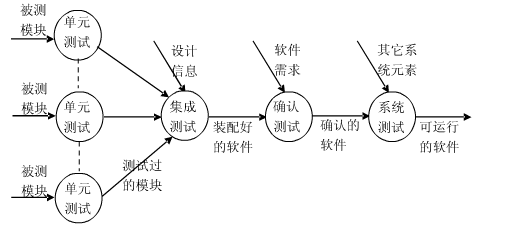
通过点击来响应操作。例如：点击按钮来进行确定，正确的执行操作。

提示和说明。针对用户的错误操作和错误的输入给出适当的提示或说明。

给出确认信息。对无法进行的操作，给予用户放弃的选择。

## 6.4测试过程

系统的测试过程是一条路漫长的路，一直延伸在系统开发的过程中。首先单元测试，在编码过程中，完成一个功能部分，就运行它，看它是否能完成自己的功能。其次是集成测试，整合每个单元，对集合的单元进行测试运行。最后进行系统测试，系统测试的目的是保证程序员实现的系统确实是用户需要的。下面详细介绍系统测试的过程与结果。

图6-1软件测试的过程

### 6.4.1功能测试

检查需求分析报告中的功能有没有被完全实现。测试时，应该已经完成基本需求分析的任务。没有完成的也要在后期完善。

### 6.4.2数据测试

单元模块测试，是测试各个模块是否可以独立运行的阶段，单元模块的测试直接影响系统整体测试的结果。

用户登录模块测试

表6-1用户登录模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 用户名长度 | 不同长度字符串 | 提示输入不正确 | 正常 |  |
| 密码 | 不同长度字符串 | 正常 | 异常 | 已处理 |
| 再次登录用户名 | 不同长度字符串 | 正常 | 正常 |  |
| 再次登录密码 | 不同长度字符串 | 正常 | 正常 |  |

用户注册模块

表6-2用户注册模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 注册用户 | 使用各种字符数字组合 | 提示输入正确 | 正常 |  |
| 重复用户 | 输入已经存在的用户名 | 提示不正确 | 正常 |  |
| 密码 | 空，长度为2，8，20，36 | 正常 | 正常 |  |

好友聊天测试：接受消息的间隔时间短，用户体验较为满意。收信息的时候有弹窗提示。可以向离线好友发送多条信息，离线好友登录后接收。

表6-3好友聊天模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 发送信息 | 不同长度字符串 | 发送成功 | 正常 |  |
| 给用户离线发送信息 | 不同长度字符串 | 发送成功 | 正常 |  |
| 给用户离线发送信息，用户登录是否有提示 | 不同长度字符串 | 发送成功，用户登录有提示 | 正常 |  |

创建班级测试：创建班级成功，创建后可以在班级搜索中搜索中，点击加入班级可以进入班级。

表6-4创建班级模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 创建班级 | 学校，类别，时间（从什么时间开始到什么时间结束），班级名，班级简介 | 创建成功 | 正常 |  |
| 创建班级后可在搜索班级中查找到班级 | 刚刚创建的班级名称 | 搜索成功 | 正常 |  |
| 加入刚刚创建的班级 | 无 | 可以申请加入班级 | 正常 |  |

搜索班级测试：搜索出相应班级信息。

表6-5搜索班级模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 根据班级名称搜索班级 | 班级名称 | 搜索成功 | 正常 |  |
| 根据学校和类型搜索班级 | 学校和班级类型 | 搜索成功 | 正常 |  |
| 根据学校，类型和班级名称搜索班级 | 学校，类型和班级名称 | 搜索成功 | 正常 |  |

发送加入班级申请：发送加入班级申请，班级创建人可以接收并处理。

表6-6发送加入班级申请模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 选择班级点击发送加入班级申请按钮 | 班级 | 申请发送成功，班级管理员接到申请 | 正常 |  |

处理加入班级申请：班级创建人处理加入班级申请，处理通过的可以加入班级，处理不通过的不可以加入班级，并给申请人反馈。

表6-7处理加入班级申请模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 通过加入班级申请 | 加入班级申请 | 申请人接收提示，加入班级成功 | 正常 |  |
| 拒绝加入班级申请 | 加入班级申请 | 申请人接收提示被拒绝，无法加入班级 | 正常 |  |

班级聊天测试：多人聊天可以实现，用户接受信息顺畅。班级中所有用户无论是否在线都会收到群消息。不在线的用户登陆后会接受信息。

表6-8班级聊天模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 在班级聊天中发送信息 | 不同长度字符串 | 发送成功 | 正常 |  |
| 在班级聊天中发送信息，班级中的在线成员可以接收 | 不同长度字符串 | 信息发送成功，在线人员接收成功 | 正常 |  |
| 在班级聊天中发送信息，班级中的不在线成员可以接收 | 不同长度字符串 | 信息发送成功，不在线人员接收成功 | 正常 |  |

动态发布，浏览与评论测试：可添加多张图片，并可以预览；可指定该动态的权限：仅自己可见、好友可见、同学可见、所有人可见。动态发布成功后，可在浏览动态模块查看

表6-9动态模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 文字动态发布 | 不同长度字符串 | 发布成功 | 正常 |  |
| 图片动态发布 | 图片 | 发布动态之前可以预览，可以发送多个图片 | 正常 |  |
| 动态发布成功后，可在浏览动态模块查看 | 不同长度字符串和图片 | 浏览成功 | 正常 |  |
| 可以在动态下评论，可以回复该评论 | 不同长度字符串和图片 | 评论成功并可以被回复 | 正常 |  |

留言发布测试：留言发布可以发送文本和图片，可以选择好友观看权限。

表6-10留言模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 公开留言发布 | 不同长度字符串 | 发布成功，在留言区大家都可以查看该条留言 | 正常 |  |
| 私密留言发布 | 不同长度字符串 | 发布成功，在留言区只有好友和自己可以查看该条留言 | 正常 |  |
| 回复好友留言 | 不同长度字符串 | 回复成功 | 正常 |  |

班级话题测试：班级成员可以在班级话题模块内按条件查询已有话题、加入某个话题进行留言，或是发起自己的话题。

表6-11班级话题模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 班级话题发布 | 不同长度字符串 | 发布成功，在话题区大家都可以查看并加入该话题 | 正常 |  |
| 查询班级话题 | 班级话题 | 查询所有话题 | 正常 |  |
| 加入话题 | 不同长度字符串 | 加入成功，可以进行留言等操作 | 正常 |  |

班级活动测试：在班级活动中，用户可以按条件查看班级活动，加入、退出班级活动，发起或取消班级活动。对于已过期的活动用户不能参加。

表6-12班级活动模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 发起班级活动 | 活动相关属性，活动主题，活动时间等 | 发起成功 | 正常 |  |
| 查询全部班级活动 | 班级活动 | 可以查询全部班级活动 | 正常 |  |
| 查询我加入的班级活动 | 班级活动 | 查询成功 | 正常 |  |
| 查询我发起的班级活动 | 班级活动 | 查询成功 | 正常 |  |
| 加入班级活动 | 班级活动 | 加入成功 | 正常 |  |
| 加入已经过期的班级活动化 | 班级活动 | 提示活动已经过期，加入失败 | 正常 |  |
| 退出我加入的班级活动 | 班级活动 | 退出活动成功 | 正常 |  |

文件收发的测试：文件收发成功，并对发送人，发送时间等进行记录。

表6-13文件收发模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 文件发送 | 文件 | 发送成功 | 正常 |  |
| 文件接收 | 文件 | 接收成功 | 正常 |  |

好友的查询 添加与删除：查询自己的好友列表。添加，删除自己的好友。

表6-14好友模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 查询好友 | 用户 | 查询成功 | 正常 |  |
| 添加好友 | 用户 | 添加信息发送成功 | 正常 |  |
| 删除好友 | 好友列表中好友 | 删除成功 | 正常 |  |

修改个人资料：修改自己的个人资料，修改头像。不可以修改用户名。

表6-16班级话题模块测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试内容 | 测试数据 | 期望结果 | 实际结果 | 处理结果说明 |
| 修改个人资料 | 性别等属性，用户名不可修改 | 修改成功 | 正常 |  |
| 修改个人头像 | 图片 | 修改成功 | 正常 |  |

### 6.4.3整体测试

综合上面的测试，各个页面的连接正常。并且对各个页面添加返回的链接。测试结果：用户页面美观，简洁，易于使用。基本满足了一个同学聚会交友系统需要的功能。系统运行稳定性较好，没有安全问题。

## 6.5测试结果

经过上面的测试，系统运行正常，且用户页面美观，简洁，易于使用。但是系统还存在某些技术问题。例如，系统运行不太稳定。软件测试可以发现一些系统的缺陷，进而可以改进我们的系统。使我们的系统更加完善，安全，用户体验更好。并且也可以让我们认识到在软件设计与开发过程中的一些不足，使我们下一次的设计可以考虑的更加全面且无误。

# 7总结

本文对所选择的开发语言做了介绍，本系统在开发过程中采用Java、Jsp、AJAX等技术。此外，在系统的开发过程中采用规范的软件工程的开发流程，因此在需求分析及数据库设计部分做了大量的工作，这些对本系统后期的编码过程起到了指导作用，在本文中也有所体现ttttttttt。

本系统也将会在今后的应用中逐步完善，做到尽善尽美。经过几个月的毕业设计，到今天已告一个段落。从开始设计，到目前为止功能基本上都已经实现了。

在设计中我深知自己掌握的知识还远远不够，掌握的一些理论知识应用到实践中去，总会出现这样或那样的问题，不是理论没有掌握好，而是光知道书本上的知识是远远不够的，一定要把理论知识和实践结合起来。把学到的知识应用到时间中去，多做多练，才可以把理论的精华发挥出来。知识不是知道，了解就好，一定要去应用它，发展它，让它在现实生活中得到充分的应用，从而解决一些问题，这才是学习的根本目的。

在设计阶段，通过对课题的深入分析与研究，迫使我对技术有了一定的了解。在遇到问题时，得到了指导老师与同学的悉心帮助，使我感受到集体的力量是无穷的。

通过这次设计，我学会了和别人配合工作，因为一个人所学的知识不可能面面俱到的，只有通过合作，发挥自己的优点，体现团队精神，才能使工作做得更为出色。通过这次设计，我学到了许多书本上学不到的知识，增强了自己的动手能力。即将毕业我十分珍惜这次锻炼的机会，我按部就班的完成了自己的设计任务，但由于自己的知识水平有限，仍然存在很多的不足之处，恳请老师多多指教！毕业设计对于我们即将离校的同学来说，是离校前很好的一次锻炼，使我们各方面的能力都有了很大的提高，为我们踏出校门，走上社会增强了能力与自信！

# 后 记

在本次毕业设计过程中，得到了张丽娜老师细心的指导与支持，帮助我顺利完这次毕业设计起到了非常关键性的作用。在此深表敬意!

大学四年中还得到众多老师的关心支持和帮助。在传授我专业知识和学科技能的同时，还以身作则的为我以后的学习和工作树立了良好的榜样，这些都使我受益匪浅。在此，我感谢学校领导和老师对我的辛苦栽培和谆谆教诲，感谢陪伴我渡过四年时光的老师和同学们。在我即将步入社会之际，肩膀上背负的责任更加重大，为此，我更会牢记师长们的教导，踏踏实实地做事，坦坦荡荡地做人，把在大学所学的知识技能，全部用到工作之中，不辜负师长们对我的期望。这篇毕业论文，从定题、写作到修改，都直接得到了刘亚丽老师的悉心指导和耐心点拨，在此深表敬意!

最后，感谢参加论文评审和答辩的各位老师。由于时间仓促及本人学识有限，文中难免有不足之处，恳请大家批评指正。

# 参考文献

[1] 陈浩. Java从入门到精通[M]. 机械工业出版社, 2011.

[2] (美）Y, Daniel, Liang. java语言程序设计-基础篇(第8版)[M]. 北京:机械工业出版社, 2015.

[3] (美）Tom Negrino, Dori Smith. JavaScript基础教程（第8版)[M]. 北京:人民邮电出版社, 2012.

[4] 李兴华, 王月清. Java Web开发实战经典[M]. 北京:清华大学出版社, 2010.

[5] 秦小波. 设计模式之禅[M]. 北京:机械工业出版社, 2014.

[6] 埃史尔. Java编程思想(第4版)[M]. 北京:电子工业出版社出版社, 2015.

[7] 霍斯特曼. Java核心技术[M]. 北京:电子工业出版社出版社, 2015.

[8]（美）Faithe WEempem. HTML5从入门到精通[M]. 北京:清华大学出版社, 2013.

[9] Bryan Basham, Kathy Sierra, Bert Bates. Head First Servlets & JSP[M]. American :O'Reilly Media, Inc., 2008.

[10] Jive Li.Rough Set Based Rule Evaluations and Their Applications[D].University of Waterloo,Canada.20014：1-10.

[11] 陈恒,张一鸣. Struts2框架应用教程 北京：清华大学出版社 2016.3

[12] 顼宇峰. Highcharts网页图表制作实例详解 北京：清华大学出版社 2016.2

[13] Ping-Feng Pai Fong-Chuan Lee.A Rough Set Based Model in Water Quality Analysis[DB/OL]. National Chi Nan University,Taiwan.2015:1-11.

[14] Lin Zhou.,Feng Jiang.A Rough Set Approach to Feature Selection Based on Relative Decision Entropy[D].College of Information Science and Technology,Qingdao University of Science and Technology,China.2015：1-9.

[15] Moraski B. Mandating Party Development in the Russian Federation: Effects of the 2001 Party Law[J]. Journal of Elections Public Opinion & Parties, 2006, 16(3):199-219.

[16] 卞明哲, 刘光昌, 黄浩文. SMS交友系统的设计与实现[J]. 计算机工程与设计, 2004, 25(12):2272-2273.

[17] 肖建东. 浅谈网络婚恋交友系统的设计与实现[J]. 计算机光盘软件与应用, 2011(21):190-190.

# 附录一：

已明确的字典项：

表1:班级类别对照表

|  |  |  |
| --- | --- | --- |
| VAL\_ID | VAL\_VALUE | DESCRIPTION |
| 1 | 小学 | 学校类型：小学 |
| 2 | 初中 | 学校类型：初中 |
| 3 | 高中 | 学校类型：高中 |
| 4 | 大学 | 学校类型：大学 |
| 5 | 其它 | 学校类型：其它 |

表2性别对照表

|  |  |  |
| --- | --- | --- |
| VAL\_ID | VAL\_VALUE | DESCRIPTION |
| 0 | 男 | 性别：男 |
| 1 | 女 | 性别：女 |

表3:用户班级角色对照表

|  |  |  |
| --- | --- | --- |
| VAL\_ID | VAL\_VALUE | DESCRIPTION |
| 1 | 学生 | 用户是班级的学生 |
| 2 | 老师 | 用户是班级的老师 |

表4:动态权限对照表

|  |  |  |
| --- | --- | --- |
| VAL\_ID | VAL\_VALUE | DESCRIPTION |
| 1 | 自己可见 | 仅用户自己可见 |
| 2 | 好友可见 | 用户及用户好友可见 |
| 3 | 同学可见 | 用户及用户所在班级成员可见 |
| 4 | 公共可见 | 全部已注册用户可见 |

表5:对申请处理结果对照表

|  |  |  |
| --- | --- | --- |
| VAL\_ID | VAL\_VALUE | DESCRIPTION |
| 0 | 不同意 | 不同意申请 |
| 1 | 同意 | 同意申请 |
| -1 | 未处理 | 申请暂未处理 |

表6:班级职务对照表

|  |  |  |
| --- | --- | --- |
| VAL\_ID | VAL\_VALUE | DESCRIPTION |
| 1 | 班长 | 班级职务：班长 |
| 2 | 学委 | 班级职务：学委 |
| 3 | 生活委员 | 班级职务：生活委员 |
| 4 | 团支书 | 班级职务：团支书 |
| 5 | 其它 | 班级职务：其它职务 |
| 6 | 无 | 班级职务：不担任职务 |

表7是否对照表

|  |  |  |
| --- | --- | --- |
| VAL\_ID | VAL\_VALUE | DESCRIPTION |
| 0 | 否 | 是否：否 |
| 1 | 是 | 是否：是 |