**xm**

**论文题目： SNS网站的设计与实现**

摘 要

刚从学校出来的我们，由于生活，没有太多的时间去找朋友，去联系朋友一起玩耍，可是我们需要交流，需要一款可以满足我们需求的产品，而且还可以向我们推荐一些志同道合的朋友，能让我们的交流变得更加有意思，让我们去认识更多的人，我们需要交流。

本产品为用户提供了添加好友，但是不支持删除好友操作、提供发表日志，但是不提供删除功能，还提供查询其他用户的日志，统计用户每天的热词，为了以后更好的推荐好友做数据铺垫，让产品更懂用户。

本产品采用了比较热门的spring，springMvc,mybatis作为网站的技术框架，采用MVC分离来设计，中间为了加快网站的响应速度减少数据库层面的压力，采用redis作为缓存，这样可以大大减少数据库的压力，减少了网络的开销，提升了用户的体验。由于用户每天产生的数据比较大，而且聊天的内容非常重要，为了更好的聊天，采用了websocket作为聊天传输的技术，这样极大的加快了传输效率，而且减少了服务器端的压力。每天还会定期去执行定时任务，将用户聊天的内容提取出来通过中文分词，放入spark进行计算统计用户的每天热词，让用户知道最近自己爱说什么话。知道自己处于什么状态。nginx应对用户量过大时候做路由分发。避免所有流量都在一台服务器上面造成压力过大。

**关键词**：朋友圈 中文分词 spark redis spring mybatis mysql websocket nginx

**Abstract**

We just came out from the school, because of life, didn't have much time to look for a friend, to contact a friend to play, but we need to communicate, need a can meet the demand of our products, but also can be recommended to us by some like-minded friends, to make our communication become more interesting, let's go to meet more people, we need to communicate.

This product provides users with add buddy, but does not support remove buddy operation, provide issued a log, but does not provide a delete function, also provide query other users log statistics every day hot word, for future better reference data, make the products more users.

This product USES the popular spring, for springMvc, mybatis as technical framework of website, MVC separation is used to design, middle in order to accelerate the speed of response of site to reduce the pressure on the level of database, using redis as caching, can greatly reduce the pressure of the database, so reduce the overhead of network and improve the user experience. The data generated every day is large, and the content of the chat is very important, in order to better chat, chat websocket is adopted as the transmission technology, this greatly speeds up the transfer efficiency, and reducing the pressure on the server side. Every day will be on a regular basis to carry out tasks regularly, extract the contents of user chat by Chinese word segmentation, the user into the spark to calculate statistics.nginx respond to users is too large when do routing distribution. Avoid all the traffic on a server cause stress.

**Keywords**: friendZero chinese spark redis spring mybatis mysql websocket nginx

**目录**

[第一章 绪 论 1](#_Toc483229341)

[1.1引言 1](#_Toc483229342)

[1.2重点 1](#_Toc483229343)

[1.3相关技术简介 2](#_Toc483229344)

[1.3.1 redis技术介绍 2](#_Toc483229345)

[1.3.2 spring框架 2](#_Toc483229346)

[1.3.3 mybatis框架 3](#_Toc483229347)

[1.3.4 spark 4](#_Toc483229348)

[1.3.5 jsp 4](#_Toc483229349)

[第二章 系统分析 5](#_Toc483229350)

[2.1 系统初步调查 5](#_Toc483229351)

[2.1.1 任务分析 5](#_Toc483229352)

[2.1.2 工作原理分析 5](#_Toc483229353)

[2.2系统功能分析 6](#_Toc483229354)

[2.2.1注册模块 6](#_Toc483229355)

[2.2.2 登录模块 6](#_Toc483229356)

[2.2.3 系统描述 7](#_Toc483229357)

[2.3 性能需求分析 8](#_Toc483229358)

[2.4数据分析 8](#_Toc483229359)

[第三章　系统设计 11](#_Toc483229360)

[3.1系统总体架构 11](#_Toc483229361)

[3.2数据库表设计 12](#_Toc483229362)

[第四章 系统实现 15](#_Toc483229363)

[4.1 数据库连接的实现 15](#_Toc483229364)

[4.3 用户功能的实现 16](#_Toc483229365)

[4.3.1 网站首页的实现 16](#_Toc483229366)

[4.3.2 注册界面的实现 17](#_Toc483229367)

[4.3.3 登录的实现 17](#_Toc483229368)

[4.3.4 用户登录后首页的实现 18](#_Toc483229369)

[4.3.5 用户好友页面的实现 19](#_Toc483229370)

[4.3.6 用户个人信息页面的实现 20](#_Toc483229371)

[4.4 管理员功能的实现 22](#_Toc483229372)

[第五章 系统测试 24](#_Toc483229373)

[总 结 26](#_Toc483229374)

[参考文献 27](#_Toc483229375)

# 

# 第一章 绪 论

## 1.1引言

每次毕业之后，身边的朋友们都各奔东西，那么我们需要怎么进行交流呢，总不能像以前一样写信吧，随着计算机的高速发展，各种各样的网络工具层出不穷，可是总有一些眼花缭乱的东西，我们只需要简单的交流就好了，然后通过其他人又去认识更多的人。

六度分隔理论：你想要与这个世界上任何一个陌生人产生联系，实际上最多只需要经过6个人。理论最早由来是一个来自哈佛的心理学教授在上个世纪60年代所做的一系列实验，当时被叫做“小世界实验”。所谓的“小世界实验”确切的说是1967年哈佛大学的心理学教授斯坦利·米尔格拉姆做过一次连锁信实验，在实验中，他将一些信件交给自愿的参加者，要求他们通过自己的熟人将信传到信封上指明的收信人手里，实验结束了之后他发现，在294封信件中有64封最终送到了目标人物手中。而在成功传递的信件中，平均只需要5.5次转发，就能够到达目标人物手中。Facebook 的此次研究是采用的米兰大学 Web 算法实验室(Laboratory for Web Algorithmics of the Università degli Studi di Milano)开发的 state-of-the-art 算法(state-of-the-art algorithms)，研究表明，用六度来描述实际中两个人之间联系的间隔稍微显得有点大，实际在Facebook上， 任何2个用户之间只有5度间隔的概率是99.6%，任何2个用户之间只有 4 度间隔的概率是92%。Facebook 的研究对象是一个月内访问 Facebook 的7.21亿活跃用户，超过世界人口的10％，所以他们的结论应该更加可信。Our4.org认为：如果研究成果可信的话，那么六度分离理论或将被改写。

但世界世界比我们想象的更小。今天社交网络巨头 Facebook和米兰大学共同宣布了他们关于六度分离理论的新研究成果：他们已经确定世界上任何两个独立的人之间平均所间隔的人数为4.74。

本产品在中学生中比较的流行，中学生比较容易去接受一些新鲜的事务，并且通过青少年去影响一些中年人，慢慢的让中年人也喜欢上这款产品。

## 1.2重点

本文讲解了社交网站的分析、设计与实现，通过对用户需求的收集与分析，确定网站的基本功能为用户信息管理部分、用户热词管理、用户空间管理。最后采用MVC模式设计了系统的架构，使用redis作为缓存，并对系统进行了简单的测试。网站的主要特点包括：

1．界面简洁大方

交互性强，操作简单

2．用户热词管理

增加用户的使用体验。

3. 系统架构

便于维护以及添加缓存加快访问速度。

4. 系统的跨平台性

由于是网页版，跟系统无关，而且轻便无需APP

5. nginx作为路由分发

避免服务器压力太大挂掉

## 1.3相关技术简介

### 1.3.1 redis技术介绍

REmote DIctionary Server(Redis) 是一个由Salvatore Sanfilippo写的key-value存储系统。Redis提供了一些丰富的数据结构，包括 lists, sets, ordered sets 以及 hashes ，当然还有和Memcached一样的 strings结构.Redis当然还包括了对这些数据结构的丰富操作。丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。原子 – Redis的所有操作都是原子性的，同时Redis还支持对几个操作全并后的原子性执行。丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性。Redis有着更为复杂的数据结构并且提供对他们的原子性操作，这是一个不同于其他数据库的进化路径。Redis的数据类型都是基于基本数据结构的同时对程序员透明，无需进行额外的抽象。Redis运行在内存中但是可以持久化到磁盘，所以在对不同数据集进行高速读写时需要权衡内存，应为数据量不能大于硬件内存。在内存数据库方面的另一个优点是，相比在磁盘上相同的复杂的数据结构，在内存中操作起来非常简单，这样Redis可以做很多内部复杂性很强的事情。同时，在磁盘格式方面他们是紧凑的以追加的方式产生的，因为他们并不需要进行随机访问。

Redis 是一个高性能的key-value数据库。 redis的出现，很大程度补偿了memcached这类key/value存储的不足，在部 分场合可以对关系数据库起到很好的补充作用。它提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。[1]

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

redis的官网地址，非常好记，是redis.io。（特意查了一下，域名后缀io属于国家域名，是british Indian Ocean territory，即英属印度洋领地）。

就DB来说，Redis成绩已经很惊人了，且不说memcachedb和Tokyo Cabinet之流，就说原版的memcached，速度似乎也只能达到这个级别。Redis根本是使用内存存储，持久化的关键是这三条指令：SAVE BGSAVE LASTSAVE …

当接收到SAVE指令的时候，Redis就会dump数据到一个文件里面。

值得一说的是它的独家功能：存储列表和集合，这是它与mc之流相比更有竞争力的地方。

不介绍mc里面已经有的东东，只列出特殊的：

TYPE key — 用来获取某key的类型

KEYS pattern — 匹配所有符合模式的key，比如KEYS \* 就列出所有的key了，当然，复杂度O(n)

RANDOMKEY - 返回随机的一个key

RENAME oldkeynewkey— key也可以改名

列表操作，精华

RPUSH key string — 将某个值加入到一个key列表末尾

LPUSH key string — 将某个值加入到一个key列表头部

LLEN key — 列表长度

LRANGE key start end — 返回列表中某个范围的值，相当于mysql里面的分页查询那样

LTRIM key start end — 只保留列表中某个范围的值

LINDEX key index — 获取列表中特定索引号的值，要注意是O(n)复杂度

LSET key index value — 设置列表中某个位置的值

LPOP key

RPOP key — 和上面的LPOP一样，就是类似栈或队列的那种取头取尾指令，可以当成消息队列来使用了

集合操作

SADD key member — 增加元素

SREM key member — 删除元素

SCARD key — 返回集合大小

SISMEMBER key member — 判断某个值是否在集合中

SINTER key1 key2 ... keyN — 获取多个集合的交集元素

SMEMBERS key — 列出集合的所有元素

还有Multiple DB的命令，可以更换db，数据可以隔离开，默认是存放在DB 0。

### 1.3.2 spring框架

Spring 框架是一个分层架构，由 7 个定义良好的模块组成。Spring 模块构建在核心容器之上，核心容器定义了创建、配置和管理 bean 的方式。

核心容器：核心容器提供 Spring 框架的基本功能。核心容器的主要组件是 BeanFactory，它是工厂模式的实现。BeanFactory 使用控制反转 （IOC） 模式将应用程序的配置和依赖性规范与实际的应用程序代码分开。

Spring 上下文：Spring 上下文是一个配置文件，向 Spring 框架提供上下文信息。Spring 上下文包括企业服务，例如 JNDI、EJB、电子邮件、国际化、校验和调度功能。

Spring AOP：通过配置管理特性，Spring AOP 模块直接将面向方面的编程功能集成到了 Spring 框架中。所以，可以很容易地使 Spring 框架管理的任何对象支持 AOP。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP，不用依赖 EJB 组件，就可以将声明性事务管理集成到应用程序中。

Spring DAO：JDBC DAO 抽象层提供了有意义的异常层次结构，可用该结构来管理异常处理和不同数据库供应商抛出的错误消息。异常层次结构简化了错误处理，并且极大地降低了需要编写的异常代码数量（例如打开和关闭连接）。Spring DAO 的面向 JDBC 的异常遵从通用的 DAO 异常层次结构。

Spring ORM：Spring 框架插入了若干个 ORM 框架，从而提供了 ORM 的对象关系工具，其中包括 JDO、Hibernate 和 iBatis SQL Map。所有这些都遵从 Spring 的通用事务和 DAO 异常层次结构。

Spring Web 模块：Web 上下文模块建立在应用程序上下文模块之上，为基于 Web 的应用程序提供了上下文。所以，Spring 框架支持与 Jakarta Struts 的集成。Web 模块还简化了处理多部分请求以及将请求参数绑定到域对象的工作。

Spring MVC 框架：MVC 框架是一个全功能的构建 Web 应用程序的 MVC 实现。通过策略接口，MVC 框架变成为高度可配置的，MVC 容纳了大量视图技术，其中包括 JSP、Velocity、Tiles、iText 和 POI。

Spring 框架的功能可以用在任何 J2EE 服务器中，大多数功能也适用于不受管理的环境。Spring 的核心要点是：支持不绑定到特定 J2EE 服务的可重用业务和数据访问对象。毫无疑问，这样的对象可以在不同 J2EE 环境 （Web 或 EJB）、独立应用程序、测试环境之间重用。

### 1.3.3 mybatis框架

mybatis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。

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。

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。

MyBatis 本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。iBATIS一词来源于“internet”和“abatis”的组合，是一个基于Java的持久层框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects（DAO）。

MyBatis 最强大的特性之一就是它的动态语句功能。如果您以前有使用JDBC或者类似框架的经历，您就会明白把SQL语句条件连接在一起是多么的痛苦，要确保不能忘记空格或者不要在columns列后面省略一个逗号等。动态语句能够完全解决掉这些痛苦。

　　尽管与动态SQL一起工作不是在开一个party，但是MyBatis确实能通过在任何映射SQL语句中使用强大的动态SQL来改进这些状况。动态SQL元素对于任何使用过JSTL或者类似于XML之类的文本处理器的人来说，都是非常熟悉的。在上一版本中，需要了解和学习非常多的元素，但在MyBatis 3 中有了许多的改进，现在只剩下差不多二分之一的元素。MyBatis使用了基于强大的OGNL表达式来消除了大部分元素。

### 1.3.4 spark

Spark让开发者可以快速的用Java、Scala或Python编写程序。它本身自带了一个超过80个高阶操作符集合。而且还可以用它在shell中以交互式地查询数据。除了Map和Reduce操作之外，它还支持SQL查询，流数据，机器学习和图表数据处理。开发者可以在一个数据管道用例中单独使用某一能力或者将这些能力结合在一起使用。在这个Apache Spark文章系列的第一部分中，我们将了解到什么是Spark，它与典型的MapReduce解决方案的比较以及它如何为大数据处理提供了一套完整的工具。

Spark通过在数据处理过程中成本更低的洗牌（Shuffle）方式，将MapReduce提升到一个更高的层次。利用内存数据存储和接近实时的处理能力，Spark比其他的大数据处理技术的性能要快很多倍。Spark还支持大数据查询的延迟计算，这可以帮助优化大数据处理流程中的处理步骤。Spark还提供高级的API以提升开发者的生产力，除此之外还为大数据解决方案提供一致的体系架构模型。Spark将中间结果保存在内存中而不是将其写入磁盘，当需要多次处理同一数据集时，这一点特别实用。Spark的设计初衷就是既可以在内存中又可以在磁盘上工作的执行引擎。当内存中的数据不适用时，Spark操作符就会执行外部操作。Spark可以用于处理大于集群内存容量总和的数据集。Spark会尝试在内存中存储尽可能多的数据然后将其写入磁盘。它可以将某个数据集的一部分存入内存而剩余部分存入磁盘。开发者需要根据数据和用例评估对内存的需求。Spark的性能优势得益于这种内存中的数据存储。支持比Map和Reduce更多的函数。优化任意操作算子图（operator graphs）。可以帮助优化整体数据处理流程的大数据查询的延迟计算。提供简明、一致的Scala，Java和Python API。提供交互式Scala和Python Shell。目前暂不支持Java。

Spark 是在 Scala 语言中实现的，它将 Scala 用作其应用程序框架。与 Hadoop 不同，Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据集。

尽管创建 Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对 Hadoop 的补充，可以在 Hadoop 文件系统中并行运行。通过名为 Mesos 的第三方集群框架可以支持此行为。Spark 由加州大学伯克利分校 AMP 实验室 (Algorithms, Machines, and People Lab) 开发，可用来构建大型的、低延迟的数据分析应用程序。

Spark Streaming：构建在Spark上处理Stream数据的框架，基本的原理是将Stream数据分成小的时间片断（几秒），以类似batch批量处理的方式来处理这小部分数据。Spark Streaming构建在Spark上，一方面是因为Spark的低延迟执行引擎（100ms+），虽然比不上专门的流式数据处理软件，也可以用于实时计算，另一方面相比基于Record的其它处理框架（如Storm），一部分窄依赖的RDD数据集可以从源数据重新计算达到容错处理目的。此外小批量处理的方式使得它可以同时兼容批量和实时数据处理的逻辑和算法。方便了一些需要历史数据和实时数据联合分析的特定应用场合。

Bagel: Pregel on Spark，可以用Spark进行图计算，这是个非常有用的小项目。Bagel自带了一个例子，实现了Google的PageRank算法。

当下Spark已不止步于实时计算，目标直指通用大数据处理平台，而终止Shark，开启SparkSQL或许已经初见端倪。

近几年来，大数据机器学习和数据挖掘的并行化算法研究成为大数据领域一个较为重要的研究热点。早几年国内外研究者和业界比较关注的是在 Hadoop 平台上的并行化算法设计。然而， HadoopMapReduce 平台由于网络和磁盘读写开销大，难以高效地实现需要大量迭代计算的机器学习并行化算法。随着 UC Berkeley AMPLab 推出的新一代大数据平台 Spark 系统的出现和逐步发展成熟，近年来国内外开始关注在 Spark 平台上如何实现各种机器学习和数据挖掘并行化算法设计。为了方便一般应用领域的数据分析人员使用所熟悉的 R 语言在 Spark 平台上完成数据分析，Spark 提供了一个称为 SparkR 的编程接口，使得一般应用领域的数据分析人员可以在 R 语言的环境里方便地使用 Spark 的并行化编程接口和强大计算能力。

### 1.3.5 jsp

JSP与PHP、ASP、ASP.NET等语言类似，运行在服务端的语言。

JSP（全称Java Server Pages）是由Sun Microsystems公司倡导和许多公司参与共同创建的一种使软件开发者可以响应客户端请求，而动态生成HTML、XML或其他格式文档的Web网页的技术标准。

JSP技术是以Java语言作为脚本语言的，JSP网页为整个服务器端的Java库单元提供了一个接口来服务于HTTP的应用程序。JSP文件后缀名为(\*.jsp)。JSP开发的WEB应用可以跨平台使用，既可以运行在Linux上也能运行在Window上。

JSP2.0中的一个主要功能是JSP fragment，它的基本特点是可以使处理JSP的容器推迟评估JSP标记属性。我们知道一般JSP是首先评估JSP标记的属性，然后在处理JSP标记时使用这些属性，而JSP fragment提供了动态的属性。也就是说，这些属性在JSP处理其标记体时是可以被改变的。JSP需要将这样的属性定义为 javax.servlet.jsp.tagext.JspFragment类型。当JSP标记设置成这种形式时，这种标记属性实际上的处理方法类似于标记体。在实现标记的程序中，标记属性可以被反复评估多次。这种用法称为JSP fragment。JSP fragment还可以定义在一个SimpleTag处理程序中使用的自制标记动作。像前面例子说明的，getJspBody返回一个 JspFragment对象并可以在doTag方法中多次使用。需要注意的是，使用JSP fragment的JSP只能有一般的文本和JSP action，不能有scriptlet和scriptlet表达式。

JSP 2.0中加入了新的创建自制标记的API，javax.servlet.jsp.tagext.SimpleTag定义了用来实现简单标记的接口。和 JSP 1.2中的已有接口不同的是，SimpleTag接口不使用doStartTag和doEndTag方法，而提供了一个简单的doTag方法。这个方法在调用该标记时只被使用一次。而需要在一个自制标记中实现的所有逻辑过程、循环和对标记体的评估等都在这个方法中实现。从这个方面来讲， SimpleTag和IterationTag可以达到同等的作用。但SimpleTag的方法和处理周期要简单得多。在SimpleTag中还有用来设置JSP内容的setJspBody和getJspBody方法。Web容器会使用setJspBody方法定义一个代表JSP内容的 JspFragment对象。实现SimpleTag标记的程序可以在doTag方法中根据需要多次调用getJspBody.invoke方法以处理JSP内容。

# 第二章 系统分析

## 2.1 系统初步调查

### 2.1.1 任务分析

用户首次需要注册，并且将用户的信息保存在了user表里面，接下来就是直接跳到登录页面，这样用户可以节省一次点击按钮的操作，接下来是对用户的密码进行加密操作，之后存入到数据库，防止数据被盗用造成不安全隐患。用户发表的信息存在graphtheory里面进行管理。

### 2.1.2 工作原理分析

网站一开始部分分为登录，注册，用户注册的时候需要去检查是否已经存在这个用户名，进行冲突性检查。登录时候去校验账号与密码的匹配性。

系统工作原理如图：



图2-1 系统工作原理图

## 2.2系统功能分析

### 2.2.1注册模块

注册账号是进入系统的必要条件，注册之后用户的基本信息会存在user表里面，接下来自动检测是否有冲突的用户名，然后就可以进入我们的系统了。用户接下来可以添加好友来进行愉快的社交之旅。



图2-2-1 用户注册模块图

### 2.2.2 登录模块

用户登录成功之后，方可进获得相对应的权限模块。



### 2.2.3 系统描述

1、注册账号

主要流程：

1.用户打开注册页面，填写用户名及密码即可，简单不复杂。

2.对数据进行校验之后即可将信息插入数据库

3.如果用户名被注册之后提示用户需要重新输入用户名

2、修改用户信息

主要流程：

1．登录进去之后首先展示的是自己的资料以及自己发表过的一些日志。

2．点击进入聊天页面之后，点击左边的侧边栏即可修改个人的信息。

3、寻找好友

主要流程：

1．用户登录之后，选择添加好友按钮之后，就可以进行查找还未添加的并且自己感兴趣的好友。

4、发表日志

主要流程：

1．用户在登录之后，可以发表自己一些想说的话，或者有想分享的东西。

2．还可以选择添加图片配上文字。

## 2.3 性能需求分析

为了让用户能够得到更好的体验，减少数据库的压力，采用了redis作为缓存来加快网站的响应速度：

1．用户角度

响应时间（最关心的指标）2/5/10原则，过长的时间等待让客户烦燥不安。

2．系统角度

网络延迟、数据延迟、系统资源的使用情况 。

3．开发角度

代码不应该有冗余，使用方法以及数据结构时候应该正确使用，不应该产生过多的变量以及不必要的GC。

## 2.4数据分析

经过需求分析及系统功能分析，本系统共有用户信息、好友信息、日志信息、聊天信息，兴趣信息，用户信息关系，联系我们，七个实体，需要对他们进行相应的数据分析与设计。

1.用户的基本信息存在自己的表里面。

2.好友的基本信息存在一张表里面，里面有好友的基本信息。

3.用户可以发表多篇自己的日志，日志可以包含图片也可以不包含图片。

4.用户之间的聊天信息存储在一张消息表里，用户之间的消息记录非常重要，必须持久化。

5.用户的兴趣也需要每天进行统计更新，这样才能掌握用户的动态信息。

6.如果用户对产品有什么不满意的可以联系我们。

7.用户与用户之间也是存在关系的，这个也存在一个表里面。

下面是相应的数据类图：



图 2-3 系统数据类图

这五个实体类的属性说明如下：

1.用户类

用户的属性为：用户名，密码，用户自我介绍，用户头像。用户名具有唯一识别性质。

2.好友类

属性为：用户名，好友名，好友自我介绍以及好友头像。

3.日志类

属性为：日志id，日志发布日期，用户名，发布内容，日志标题。

4.消息类

属性为：用户名，好友名，已读未读标志，消息具体内容，发送的时间。

5.图片类：

属性为：图片的标题，图片的内容，图片，用户名。

6.用户兴趣类

属性为:兴趣用户，热词，发布时间，最后一次更新时间。

# 第三章　系统设计

## 3.1系统总体架构

清晰的角色划分：控制器(controller)、验证器(validator)、命令对象(command obect)、表单对象(form object)、模型对象(model object)、Servlet分发器(DispatcherServlet)、处理器映射(handler mapping)、试图解析器(view resoler)等等强大而直接的配置方式：将框架类和应用程序累都能作为JavaBean配置，支持跨多个context的引用，例如，在web控制器中对业务对象和验证器validator)的引用。强大而直接的配置方式：将框架类和应用程序累都能作为JavaBean配置，支持跨多个context的引用，例如，在web控制器中对业务对象和验证器validator)的引用。可适配、非侵入：可以根据不同的应用场景，选择何事的控制器子类(simple型、command型、from型、wizard型、multi-action型或者自定义)，而不是一个单一控制器(比如Action/ActionForm)继承。可重用的业务代码：可以使用现有的业务对象作为命令或表单对象，而不需要去扩展某个特定框架的基类。可定制的handler mapping和view resolution：spring提供从最简单的URL映射，到复杂的、专用的定制策略。与某些web MVC框架强制开发人员使用单一特定技术相比，Spring显得更加灵活。Spring Bean的生命周期可以被限制在当前的HTTp Request或者HTTp Session。准确的说，这并非Spring MVC框架本身特性，而应归属于Spring MVC使用的WebApplicationContext容器。

## 3.2数据库表设计

根据需求分析，要完成整个系统的数据维护，需要建立七张表：

表1 用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段大小 | 是否允许为空 | 是否是主键 |
| userId | Varchar | 50 | 否 | 是 |
| userPaddword | Varchar | 50 | 否 | - |
| userName | Varchar | 50 | 是 | - |
| userHeadPortraitpath | Varchar | 30 | 是 | - |
| userIntroduce | Int | 4 | 是 | - |

将用户信息表保存为user，用来存储用户的基本信息。

表2 好友信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段大小 | 是否允许为空 | 是否是主键 |
| userName | Varchar | 50 | 否 | 否 |
| friendName | Varchar | 50 | 否 | 否 |

将好友表保存为friend，用来存储用户之间的好友信息。

表3 空间日志表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段大小 | 是否允许为空 | 是否是主键 |
| Id | Varchar | 50 | 否 | 是 |
| LogDayContext | Varchar | 50 | 否 | - |
| LogDayTittle | varchar | 50 | 否 | - |
| LogDayDate | datetime | - | 否 | - |
| userName | varchar | 50 | 否 | - |

日志表名为logday，以此存储所有用户发表的日志。

表4 兴趣表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段大小 | 是否允许为空 | 是否是主键 |
| id | varchar | 50 | 否 | 是 |
| name | varchar | 50 | 否 | - |
| Hot\_word | varchar | 50 | 否 | - |
| Fir\_time | Datetime |  | 否 | - |
| Last\_up\_time | Datetime |  | 否 |  |

回复表名为interest，存储用户的兴趣信息。

表5 图片表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段大小 | 是否允许为空 | 是否是主键 |
| grapethroytittle | Varchar | 50 | 否 | 是 |
| grapethroycontext | Varchar | 50 | 否 | 是 |
| grapethroyimagepicpath | Varchar | 50 | 是 | - |
| userName | Varchar | 50 | 否 | - |

照片表名为grapethroy，存储用户的图片信息。

表6 聊天信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段大小 | 是否允许为空 | 是否是主键 |
| text | varchar | 50 | 否 |  |
| sendDate | Datetime | 50 | 否 | - |
| userName | varchar | 50 | 否 | - |
| friendName | varchar | 50 | 否 | - |
| read | varchar | 50 | 否 |  |

# 第四章 系统实现

## 4.1 数据库连接的实现

使用的是mysql，用mybatis实现持久化。

部分整合代码如下:

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:context="http://www.springframework.org/schema/context"

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

http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.2.xsd

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

http://www.springframework.org/schema/aop/spring-aop-3.2.xsd

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

http://www.springframework.org/schema/tx/spring-tx-3.2.xsd ">

<context:property-placeholder location="classpath:db.properties" />

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"

destroy-method="close">

<property name="driverClassName" value="${jdbc.driver}" />

<property name="url" value="${jdbc.url}" />

<property name="username" value="${jdbc.username}" />

<property name="password" value="${jdbc.password}" />

<property name="maxActive" value="30" />

<property name="maxIdle" value="5" />

</bean>

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<!-- 数据库连接池 -->

<property name="dataSource" ref="dataSource" />

<!-- 加载mybatis的全局配置文件 -->

<property name="configLocation" value="classpath:sqlMap/sqlMapConfig.xml" />

<property name="mapperLocations">

<list>

<value>classpath:com/callRing/\*\*/mapper/\*.xml</value>

</list>

</property>

</bean>

<!-- mapper扫描器 -->

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<!-- 扫描包路径，如果需要扫描多个包，中间使用半角逗号隔开 -->

<property name="basePackage" value="com.callRing.\*.mapper"></property>

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />

</bean>

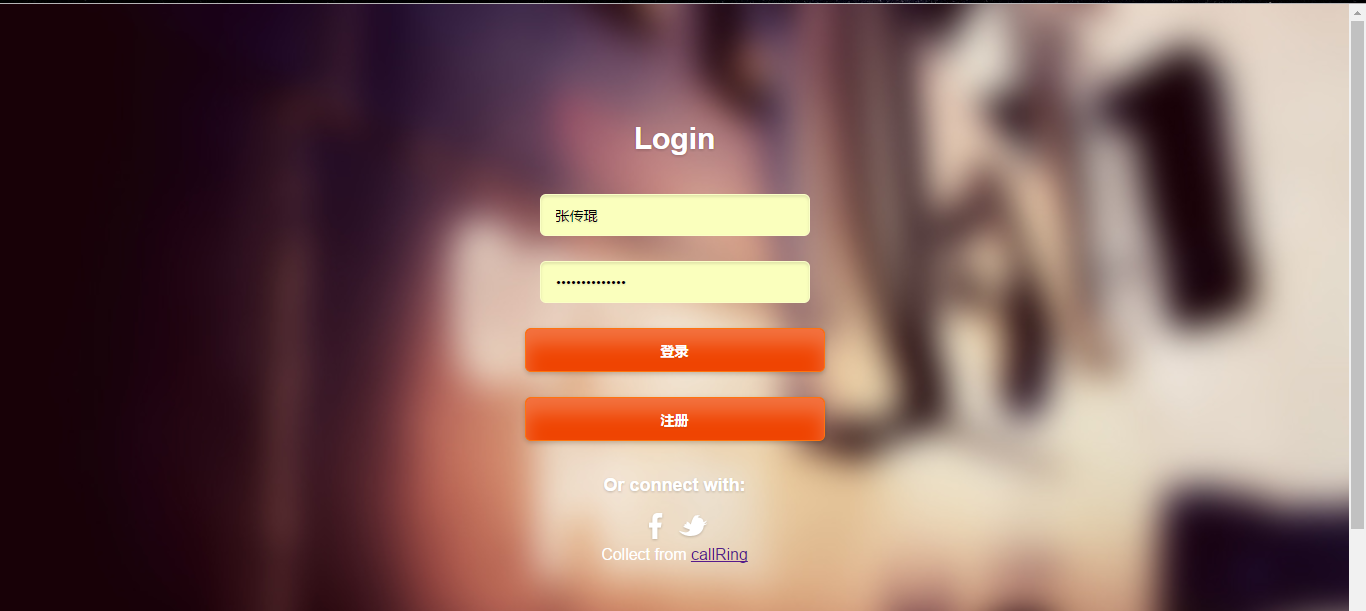
</beans>

## 4.3 用户功能的实现

### 4.3.1 网站首页的实现

系统的首页

在用户进入网站后首先会来到该界面，主页提供登陆、注册及相关链接功能。通过该页面用户可以体验到该网站提供的所有功能。

图 4-1 网站登陆页面

### 4.3.2 注册界面的实现

新用户必须要注册自己的账号，首先要填写自己的用户信息，注册信息包括：账号名、密码、真实姓名、性别、生日、身份等。其中账号和密码不能为空，其他选项可以忽略，提交系统对账号进行重复性检查，注册界面如图所示：

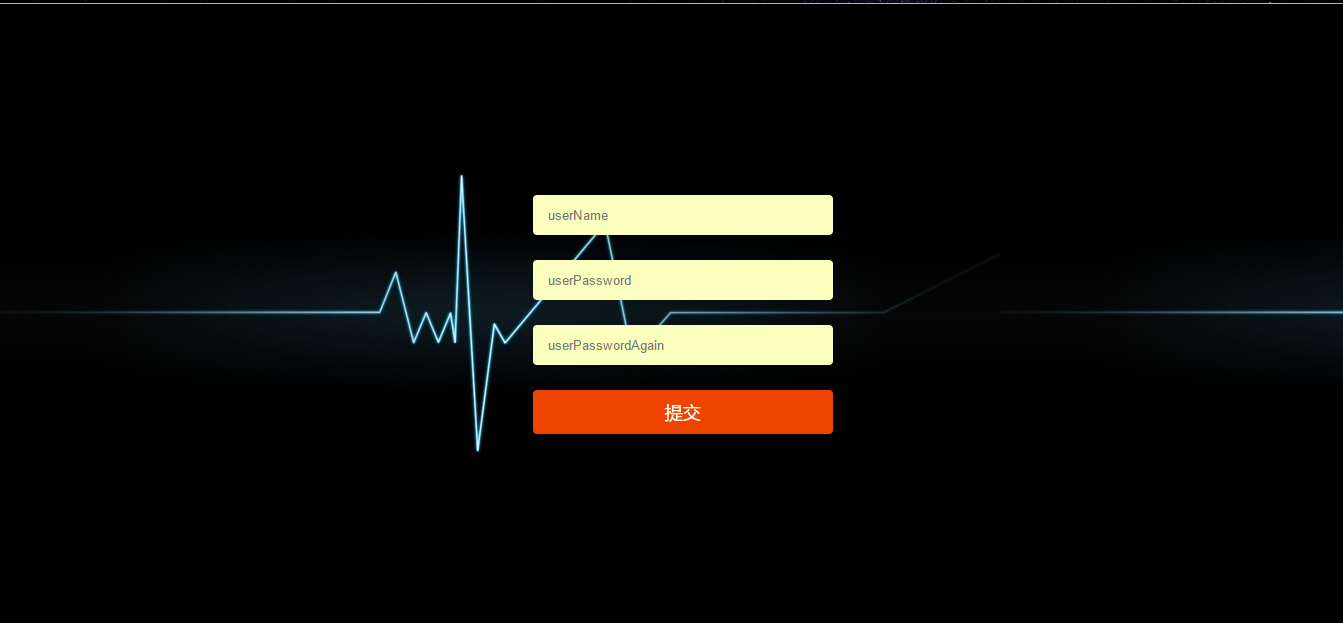
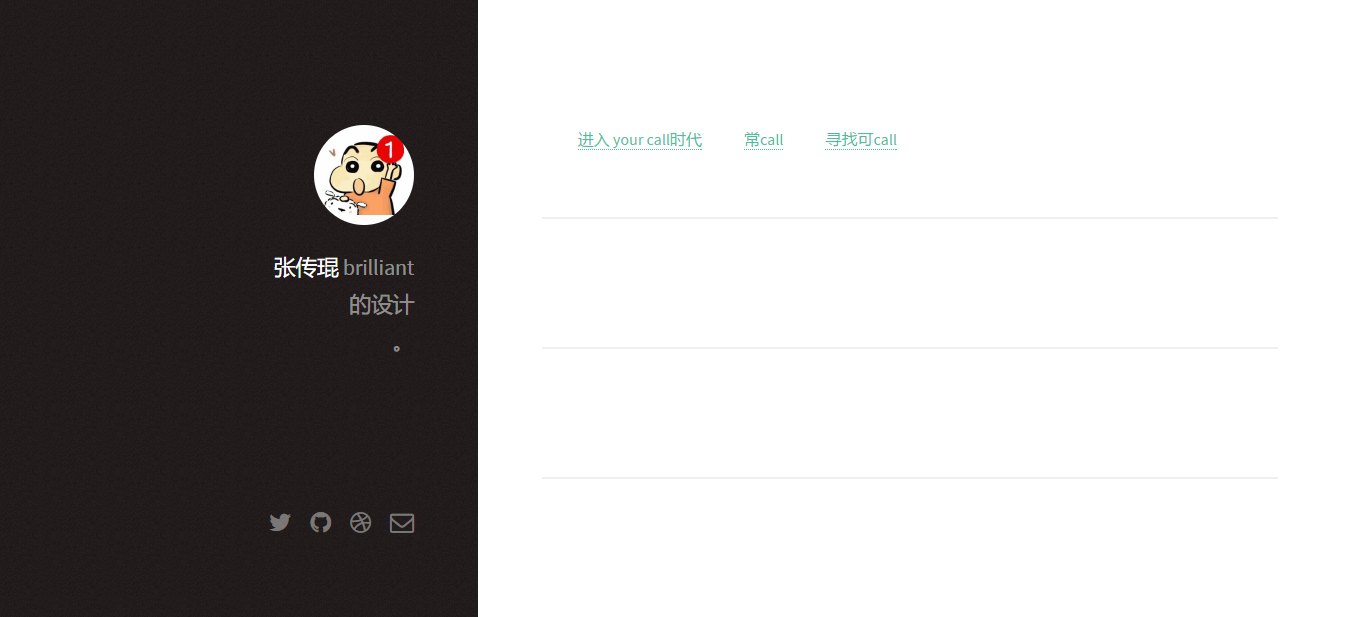


图 4-2 用户的注册界面

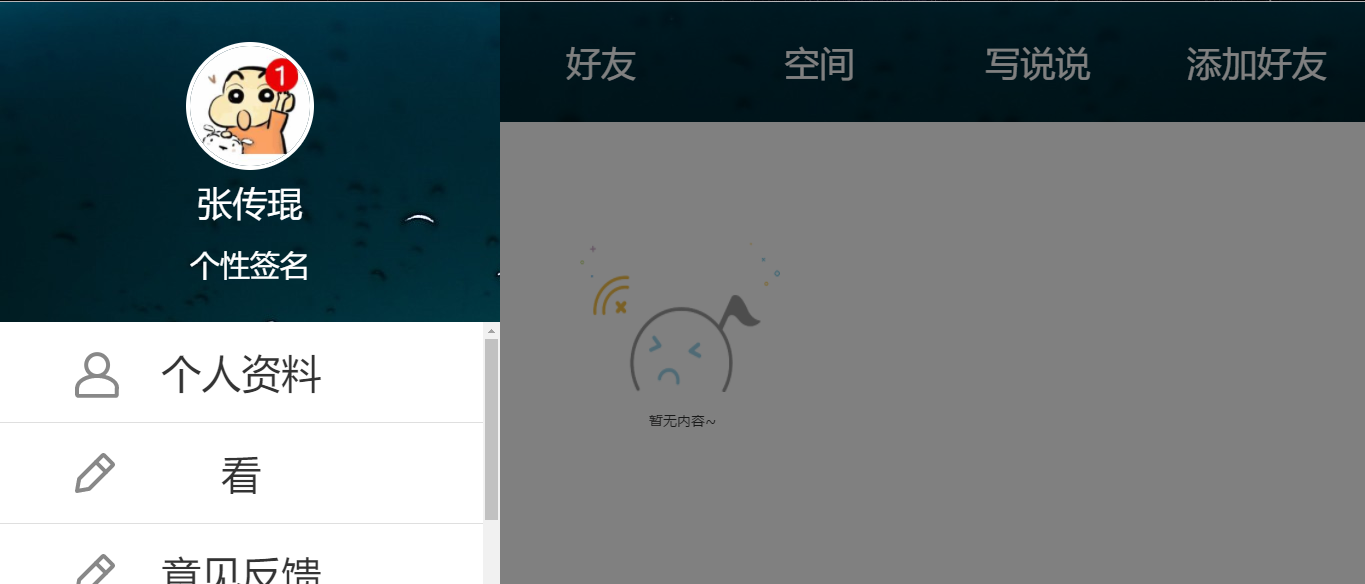
### 4.3.3 登录的实现

校验通过之后进入到用户主页。



### 4.3.4 用户登录后首页的实现

该页面用户可以查看自己发表的图片以及进入到聊天页面。



### 4.3.5 用户好友页面的实现

好友管理功能的实现主要在用户的好友界面上，这里提供的关于的好友信息所有处理。



图 4-5 好友界面



添加兴趣话题页面，此页面是用户的当天的聊天热词，每天通过定时任务凌晨去统计用户的好友聊天热词状况。

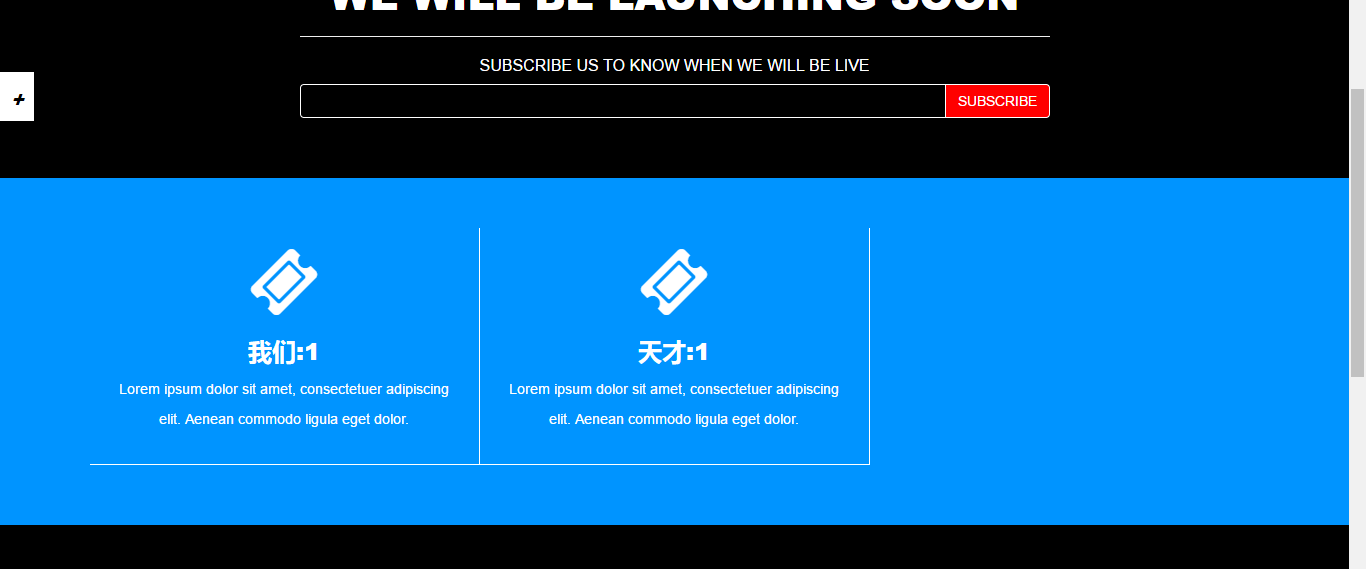


图 4-6 列出所有好友界面

### 4.3.6 用户个人信息页面的实现

图显示了用户个人信息页面的主要框架，上方是大功能导航条，左侧是个人信息的各个功能切换的导航条。这里主要实现的功能有个人信息管理、上传图片管理、个人日志管理。

上传图片需要用户给图片写上标题（可以重复），上传的图片存储到服务器的硬盘里，数据库存储着该图片的路径。个人主页中上传图片界面如下：



图 4-7 个人主页界面

# 第五章 系统测试

测试1:账号登录验证：

描述：在普通用户或者管理员登录时，用户名或密码有一项为空或者用户名不存在或者密码错误。

预测结果：应拒绝登录并出现相应的错误提示。

操作：用户名、密码任意一项为空或者用户名不存在或者密码错误。

结果：拒绝登录并都出现相应的错误提示。

测试2: 注册账号测试：

描述：注册账号是不按照指定的规则，用户名重复或者用户名密码不填或者生日格式不正确。

预测结果：注册失败并出现相应的错误提示。

操作：打开注册页面，使用一个已经存在的账号名注册。

结果：提示该账号已经存在。

测试3: 发表日志测试：

描述：用户进入到首页后写下日志并发表，检测能否发表日志。

预测结果：发表的日志应立即出现在发表日志的页面上。

操作：用户进入到首页后写下日志并发表。

结果：发表日志后，该页面自动刷新，新日志出现在页面上。

测试4: 查找好友测试：

描述：用户进入到好友页面后，输入好友名字，搜索好友。

预测结果：数据库中该名字的用户信息被显示出来。

操作：用户进入到好友页面，输入好友名字，选择完全匹配，点击搜索。

结果：显示出该好友的信息。

测试5: 删除好友测试：

描述：用户进入到好友页面，点击所有好友，选择删除，检测能否正确删除好友关系。

预测结果：删除后的好友列表里应该没有刚删除的好友信息。

操作：用户进入到好友页面，点击所有好友，选择删除。

结果：所有好友界面自动刷新，好友列表里没有刚刚删除的好友信息。

测试6: 删除用户测试：

描述：管理员删除某个用户后，使用该用户账号登录。

预测结果：拒绝登录显示账号不存在。

操作：管理员查询某个用户，将其从数据库中删除，使用该用户账号登录。

结果：拒绝登录显示用户不存在。

以上的测试结果表明，该网站系统可以正常运作，可以为用户提供较为完善的服务。

# 总 结

感谢计算机院所有的领导和老师们，感谢你们在此期间为我们付出的一切，你们教给了我们最先进的知识、教会了我们最深刻的人生哲理，你们是我们人生转折点上的启明者、是指路人。

这里要特别感谢郑山红老师。在长达半年的毕业设计中，老师给予了我非常专业、广泛、精确的指导。大到整个系统的功能设计，小到毕业论文的排版格式及网页的布局颜色，赵辉老师都会很耐心详细的讲解，帮助我解决了毕业设计中从分析设计到编码实现大大小小的各种难题。赵辉老师精深的知识水平和认真负责的工作态度深深的影响了我，即扩展了我的知识面深化了知识点，又使我对对人生态度有了新的认识，感谢赵辉老师。

最后祝亲爱的母校长春工大明天更美好，祝在母校的所有领导和老师身体健康，万事如意。

# 参考文献

[1]（美）Grady Booch,James Rumbaugh,Ivar Jacobson,邵维忠等译.UML用户指南.机械工业出版社,2007

[2]邹竹彪 JSP宝典(附盘) 电子工业出版社(2007-01出版)

[3]张海藩,软件工程(第1版),人民邮电出版社,2005

[4]毕业设计（论文）规范化要求,长春工业大学,2004.5

[5]邱郁惠,UML和OOAD快速入门,机械工业出版社,2010

[6]Donahue，J.O著,甄广启等译,jsp程序设计宝典，电子工业出版社

[7]Grady Booch著,冯博琴 冯岚 薛涛等译,面向对象分析与设计,机械工业出版社 2003

[8]Erich Gamma Richard helm著,李英军，蔡敏等译.可复用面向对象软件的基础.电子工业出版社,2009

[9]徐宝文 周毓敏 卢红敏编著,UML于软件建模,清华大学出版社

[10]丁振凡编著,Java语言实用教程,北京邮电大学出版社

[11]毋国庆 梁正平 袁梦霆 李永华编著，软件需求工程，机械工业出版社

[12]张银鹤 冉小旻 刘治国编著，JSP完全学习手册，清华大学出版社

[13]廖若雪 编著，JSP高级编程，机械工业出版社，2001-2003

[14]章立民研究室，SQL Server2005数据库开发实战，机械工业出版社，2007

[15]梁冰 陈丹丹 苏宇 编著，SQL语言参考大全，人民邮电出版社，2008