分类号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 密级\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

UDC \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 学号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

毕业论文

论文题目 基于SSM的仿QQ空间系统的设计与实现

thesisTopic Design and implementation of the imitative QQ zone system based on the SSM

学生姓名 吉王超

学 号 2013006183

所在院系 软件工程

专业班级 软件1330班

导师姓名职称 孟东霞 副教授

完成日期 2017 年 6 月 5号

2017年 6 月 5 日

太　原　理　工　大　学

毕业设计（论文）任务书

|  |
| --- |
| 毕业设计（论文）题目：  基于SSM的仿QQ空间系统的设计与实现  毕业设计（论文）要求及原始数据（资料）：  1．设计基于SSM的仿QQ空间系统；  2．实现QQ空间系统所需的功能；  3．按照典型软件工程的流程、规范和方法，完成仿QQ空间发布消息、发布日志、好友评论等功能的分析、设计、实现、测试和部署；  4．针对特定的技术或算法，给出较为深入的研究分析结果；  5．训练检索文献资料和利用文献资料的能力；  6．训练撰写技术文档与学位论文的能力。 |

|  |
| --- |
| 毕业设计（论文）主要内容：  1．研究基于SSM的仿QQ空间系统，设计模块的功能和非功能需求，并形成需求说明书。本模块应至少实现发布消息、发布日志、好友评论等功能；  2．设计系统的原型并形成原型设计文档；  3．设计系统的体系架构、数据库、开发框架、数据库、关键算法等并形成系统设计文档。  4．模块代码开发及单元测试并形成最终系统实现；  5. 系统集成、功能测试及形成测试报告；  6．编写系统安装使用文档；  7．在该系统基础上，加入一定的技术研究型内容，并最终形成毕业论文。  学生应交出的设计文件（论文）：  1．内容完整、层次清晰、叙述流畅、排版规范的毕业设计论文；  2．包括毕业设计论文、源程序等内容在内的毕业设计电子文档及其它相关材料。 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 主要参考文献（资料）：   1. 李雪莲,办公自动化系统分析与开发[D],吉林:吉林大学,2010 2. 刘凯立,SSH技术研究及其在办公自动化系统中的应用[D],北京:中国地质大学,2010 3. 欧海华,安全高效的中小企业办公自动化系统设计[D],重庆:重庆大学,2008 4. 朱雪琴,常建功,Java Web项目开发案例导航[M],北京:电子工业出版社,2012 5. 杨春花,Spring攻略[M],人民邮电出版社,2009 6. 孙更新等,Java程序开发大全[M],中国铁道出版社,2010 7. 杨少波,顾益军,J2EE项目实训Spring框架技术[M],清华大学出版社,2008 8. 米奕杭,基于SSH框架的网络购物电子商务平台的设计与实现[D],四川:四川师范大学,2010 9. 夏昕,深入浅出Hibernate[M],电子工业出版社,2010 10. 刁成嘉,UML系统建模与分析设计[M],北京:机械工业出版社,2007 11. 易琴,重庆职业学院办公自动化系统的设计与实现[D],重庆:重庆职业学院,2008 12. Seth Ladd,Expert Spring MVC and Web Flow[M],人民邮电出版社,2008 13. Paul Deck,Spring MVC学习指南[M],人名邮电出版社，2015 14. Praveen.Gupta,Spring Web MVC Framework for rapid open source J2EE application development: a case study[J],International Journal of Engineering Science and Technology, 2010  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 专业班级 | 软件1330班 | | | | | 学生 | 吉王超 | | 要求设计（论文）工作起止日期 | | | | | 2017年3月13日~2017年6月25日 | | | | 指导教师签字 | | 倪承君 | | | | 日期 | 2017年3月13日 | | 教研室主任审查签字 | | | |  | | 日期 |  | | 系主任批准签字 | | |  | | | 日期 |  | |

# 基于SSM的仿QQ空间系统的设计与实现

摘 要

QQ空间（Qzone）是腾讯公司于2005年开发出来的一个具有个性空间，具有博客（blog）的功能，自问世以来受到众多人的喜爱。在QQ空间上可以书写日志，写说说，上传用户个人的图片，听音乐，写心情，通多多重方式展现自己。除此之外，用户还可以根据个人的喜爱设定空间的背景、小挂件等，从而使每个空间都有自己的特色。截止2016年底，QQ空间月活跃账户数达到6.25亿，QQ空间智能终端月活跃账户数达到5.96亿。

本系统基于SSM的框架，对腾讯QQ空间系统进行部分功能的模仿，将主要实现以下功能：发表说说功能，留言板功能，发表日志功能，评论，上传照片，好友查询，好友添加，好友列表分组展示，移动好友分组等功能。

此次设计仿照腾讯QQ空间，虽然只能实现部分功能，但意在理解QQ空间的基本运作流程，尝试一些之前没有做过的系统。在体验QQ空间带给用户的特色功能之外，更深层次的去思考一些

关键词：SSM；仿QQ空间；好友列表分组

# **Design and implementation of the imitative QQ zone system based on the SSM**

**Abstract**

QQ space (Qzone) is a personal space developed by tencent in 2005, which has been enjoyed by many people since its inception. On the QQ space can write the journal, write to say, upload the picture of user individual, listen to music, write the mood, the multiplicity of the way show oneself. In addition, user also can set the background of the space according to individual love, small hang piece, so that each space has its own characteristic. As of the end of 2016, the number of active accounts in the QQ space month reached 625 million, and the number of active accounts in the QQ space intelligence terminal reached 596 million.

This system based on the framework of SSM, part function to tencent QQ space system imitation, will implement the following main functions: the published about function, guestbook, logging, published comments, upload photos, good friend query, add friends, friends list group show, mobile group, and other functions.

This design to simulate tencent QQ space, although only part of the function to achieve, but to understand the QQ space of the basic operation process, try some before didn't do it. To experience the features of the QQ space, we should think a little more deeply.

**Key words:** SSM; Imitation QQ space; Buddy list grouping

目 录

[基于SSM的仿QQ空间系统的设计与实现 i](#_Toc20402)

**[Design and implementation of the imitative QQ zone system based on the SSM](#_Toc4600)** [ii](#_Toc4600)

[1 绪论 1](#_Toc24266)

[1.1 开发背景和意义 1](#_Toc1109)

[1.2 开发环境 1](#_Toc23265)

[1.2.1 开发语言 1](#_Toc37)

[1.2.2 开发工具 2](#_Toc23117)

[1.2.3 开发方式 3](#_Toc24555)

[2 需求分析 4](#_Toc32002)

[2.1 可行性研究分析 4](#_Toc7512)

[2.2 功能需求分析 5](#_Toc4404)

[2.2.1 主要术语 5](#_Toc17398)

[2.2.2 目标与范围 6](#_Toc6627)

[2.2.3 主要接口 7](#_Toc16885)

[2.2.4 功能需求与非功能需求 7](#_Toc12003)

[3 总体设计 9](#_Toc20632)

[3.1 系统框架设计 9](#_Toc10794)

[3.2 数据库设计 9](#_Toc14550)

[3.3 系统主要模块设计 16](#_Toc29786)

[3.3.1 个人档 16](#_Toc7717)

[3.3.2 个人中心 16](#_Toc4843)

[3.3.3 相册 16](#_Toc2194)

[3.3.4 留言板 16](#_Toc27749)

[3.3.5 日志 17](#_Toc28808)

[3.3.6 说说 17](#_Toc7034)

[3.3.7 好友 17](#_Toc25087)

[4 详细设计和实现 18](#_Toc12306)

[4.1 系统主要程序设计与实现 18](#_Toc14819)

[4.1.1 好友 18](#_Toc30078)

[4.1.2 说说 29](#_Toc22517)

[4.1.3 相册 30](#_Toc20328)

[4.1.4 留言板 30](#_Toc12406)

[4.1.5 日志 30](#_Toc2887)

[5 系统测试与维护 32](#_Toc17493)

[5.1 软件测试 32](#_Toc12886)

[5.1.1 登录验证测试 32](#_Toc31682)

[5.1.2 注册功能测试 33](#_Toc2146)

[5.1.3 好友功能测试 34](#_Toc32330)

[5.1.4 说说发表测试 38](#_Toc1514)

[5.1.5 相册上传照片测试 39](#_Toc15598)

[5.1.6 留言板发表测试 40](#_Toc20586)

[5.1.7 日志测试 40](#_Toc17871)

[结论 42](#_Toc2500)

[参考文献 43](#_Toc30313)

[致谢 44](#_Toc22560)

[外文原文 45](#_Toc2162)

[中文翻译 57](#_Toc20937)

# 1 绪论

## 1.1 开发背景和意义

随着社会经济的迅速发展和科学技术的全面进步，计算机事业的飞速发展，以计算机与通信技术为基础的信息系统正处于蓬勃发展的时期。随着经济文化水平的显著提高，人们对于生活质量及工作环境的要求也越来越高。日常生活的社交已不能满足人们的交际圈，大家都开始进军网络世界的社交，距离在网络社交上已不再成为问题。在网络社交中,青少年网民使用即时通信软件的比例最高。QQ空间的应运而生成为了很多80,90甚至00后的必备工具，在QQ空间上可以书写日志、写说说，上传用户个人的图片，听音乐，写心情，通过多种方式展现自己。除此之外，QQ空间还为用户提供了很多高级的功能，用户可以根据自己的喜爱，来打造属于自己特色的个人主页。

本次设计的仿QQ空间系统，通过仿照实现QQ空间的部分功能来体会QQ空间原创团队的开发流程，激发学习兴趣，了解QQ空间的设计与实现。

本产品主要实现以下功能。用户登录之后可以发表说说，发表日至，上传自己的照片，搜索添加好友，及管理好友，管理分组信息等，并可实现好友评论，互动留言等功能。

## 1.2 开发环境

集成开发环境（IDE，Integrated Development Environment ）是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具。集成了代码编写功能、分析功能、编译功能、调试功能等一体化的开发软件服务套。所有具备这一特性的软件或者软件套（组）都可以叫集成开发环境。

### 1.2.1 开发语言

Java是一门面向对象的编程语言，java作为静态面向对象语言的代表，极好的实现了面向对象的理论，允许程序员以优雅的思维方式进行复杂的编程。

Java的设计者曾说过，设计者们语言的灵感主要来自于C++，整个Java语言的发展历史就是一部对C++的填坑史，Java不仅吸收了C++语言的各种优点，还摒弃了C++里难以理解的多继承、指针等概念。

20世界90年代，硬件领域出现了单片机计算机系统，Sun公司为了抢占市场先机，在1991年成立了一个Green的项目小组，专供计算机在家店产品上的嵌入式应用。Green小组首先考虑采用C++语言来编写程序，后由于C++语言的复杂和庞大，该小组决定改造C++语言，开发出一种称为Oak的面向对象语言，但由于硬件厂商出于对风险的考虑，该语言被搁置。直至1995年互联网的蓬勃发展让Oak再度崛起。Sun公司首先推出了可以嵌入网页并且可以随同网页在网络上传输的Applet，并将Oak命名为Java（注册商标时Oak已被注册，Java取自咖啡之意）。5月23日，在一个Sun World会议上，Sun公司正式发布Java和HotJava浏览器，至此Java正式诞生，随后，Java开发蓬勃发展。

Java具有简单性、面向对象、分布性、编译和解释性、稳健性、安全性、可移植性、高性能、并发性、动态性等特点。

### 1.2.2 开发工具

本次毕设所用的开发工具采用Eclipse，Eclipse 是一个开放源代码的、基于Java的可扩展开发平台，虽然其本身只是一个框架和一组服务，但它附带了一个标准的插件集，包括Java开发工具（Java Development Kit，JDK）。

Eclipse是由IBM公司投资4000万美元开发的继承开发工具，它基于Java语言编写，并且是开放源代码的、可扩展的，也是目前最流行的Java集成开发工具之一。另外，IBM公司捐出Eclipse源代码，组建了Eclipse联盟，由该联盟负责这种工具的后续开发。Eclipse为编程人员提供了一流的Java程序开发环境，它的平台体系结构是在插件概念的基础上构建的，插件是Eclipse平台最具特色的特征之一，也是其区别于其他开发工具的特征之一。

Eclipse是著名的跨平台的自由集成开发环境（IDE）。最初主要用来Java语言开发，通过安装不同的插件Eclipse可以支持不同的计算机语言，比如C++和Python等开发工具。Eclipse的本身只是一个框架平台，但是众多插件的支持使得Eclipse拥有其他功能相对固定的IDE软件很难具有的灵活性。许多软件开发商以Eclipse为框架开发自己的IDE。

虽然大多数用户很乐于将 Eclipse 当作 Java 集成开发环境（IDE）来使用，但 Eclipse 的目标却不仅限于此。Eclipse 还包括插件开发环境（Plug-in Development Environment，PDE），这个组件主要针对希望扩展 Eclipse 的软件开发人员，因为它允许他们构建与 Eclipse 环境无缝集成的工具。由于 Eclipse 中的每样东西都是插件，对于给 Eclipse 提供插件，以及给用户提供一致和统一的集成开发环境而言，所有工具开发人员都具有同等的发挥场所。

### 1.2.3 开发方式

本系统采用SSM后台框架进行开发，采用spring注入的方式结合SpringMVC+Mybatis框架进行后台设计，前台采用JQuery+boostrap框架进行设计。

# 2 需求分析

## 2.1 可行性研究分析

系统概述：本系统的用户针对80、90及00后青少年群体，致力于为用户提供一个专属于自己的个性化家园，让用户可以不受距离限制，在网上和自己的好友进行互动，实时分享自己和了解好友的最新动态。用户在首页登录后，进入查看动态页面，默认显示所有好友及自己的动态，用户可以通过导航菜单点击跳转到查个人信息页面，管理个人详细信息，并可发表说说、发表日志、上传照片。用户还可以浏览好友的空间并留言，查看他的说说列表和日志列表为他留言。同时还为用户提供了搜索好友功能，用户可搜索查看要添加的其他用户信息，并发送添加请求来添加用户，并管理自己的好友及分组信息。

本系统的主要功能如下图2.1所示

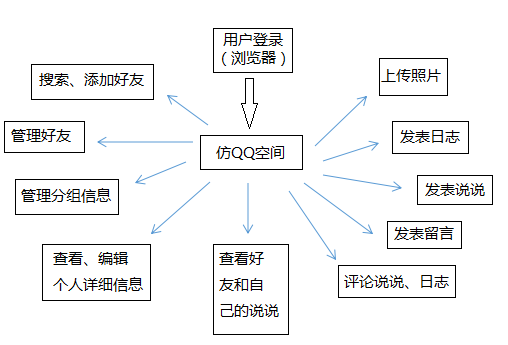


图2.1 业务流程图

技术可行性：本系统采用SSM后台框架进行开发，前台利用JQuery和bootstrap前台框架搭建，通过发布，部署在tomcat7上。系统现阶段的发展过程中，利用现有人力和物力，基于目前所掌握的技术，以上功能基本可以实现，日后的发展空间大，实现方法逐步简单容易。

操作可行性：项目启动后，用户只需要在浏览器上打开地址：输入账号密码即可开始使用本系统。由于本系统是仿照QQ空间进行开发的部分常用功能，而官方QQ空间系统自2005年出现至今已有十多年，用户群也非常庞大，大家都很熟悉QQ空间。故用户操作非常熟悉。

社会可行性：该软件只是提供信息交流平台，且交流的信息单纯，不涉及政治反动，不存在盈利和商业诈骗，没有侵犯版权，且本系统只在线下做技术交流使用，因此具有法律可行性。

经济可行性：该软件并非收益性质而开发，只是做技术交流而存在，并不承担用户交易作用，所以开发成本不会太高，故具有经济可行性。

工作负荷：由于硬件和技术有限，对于高并发的数据，系统工作负荷会过大

硬件设备：Intel（R）Core（TM）i5-3230M CPU @2.6GHz笔记本一台，运行内存4G，外存500G

软件设备：windows7及以上操作系统，Java 开发环境：jdk7，tomcat7及以上。IE8及以上。

局限性：硬件、软件、技术的局限性，使本系统只能做技术交流使用。

## 2.2 功能需求分析

随着人民生活质量的提升和现代互联网通信技术的愈发成熟和普及，人们的交际圈也开始从大街小巷开始逐步蔓延到网络上，远距离的亲朋好友也变得近在咫尺，只要有网就可以保持联系。距离不再是问题的同时，对于外面世界的新奇东西也慢慢的变得触手可及，人们也逐渐在网络上添加陌生人来扩大自己的交际圈。QQ空间的应运而生，在为用户分享好友及自己的最新动态的同时，还为用户打造了一个制作专属于自己的个人主人的功能，用户可在QQ空间里上传自己的精彩瞬间的照片保存起来，方便随时查看。

### 2.2.1 主要术语

SSM：Spring-SpringMVC-Mybatis框架

Spring是一个开源框架， Spring 是于 2003 年兴起的一个轻量级的 Java 开发框架，由 Rod Johnson 在其著作 Expert One-On-One J2EE Development and Design 中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。 Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。然而， Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。 简单来说， Spring 是一个轻量级的控制反转（ IoC ）和面向切面（ AOP ）的容器框架。

Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。SpringMVC分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

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。MyBatis 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消除了几乎所有的 JDBC 代码和参数的手工设置以及结果集的检索。MyBatis使用简单的XML或注解用于配置和原始映射，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。

JQuery： 是一个JavaScript函数库。支持HTML元素选取和操作、CSS操作、JS特效与动画、DOM处理、Ajax等。导入jQuery就像导入一段外部js代码一样简单，可以使用MS或Google的CDN进行加载。

Boostrap：BootStrap是基于HTML、CSS和JavaScript的框架，使你只需要写简单的代码就可以很快的搭建一个还不错的前端框架，他是后端程序员的福音，使他们只需要专注业务逻辑，而无须浪费太多的精力在界面设计上。它可以开发全响应式网页——不论你使用手机、平板电脑、普通个人电脑浏览网站内容，所有的元素都可以很优雅的呈现。所以，可以用他来开发适合各种设备浏览的页面，避免了大量的因为兼容性而导致的重复劳动。它的最新版本是4.0，国内目前用的较多的是3.x,我们的教程也基于BootStrap3.x版本，也是目前最受欢迎的前端框架之一。Bootstrap允许开发人员使用默认的样式和组件，也可以对其进行二次开发，当然也可以基于Bootstrap环境使用自己写的组件。

### 2.2.2 目标与范围

本系统的设计目标是为用户提供一个信息交流平台，用户可以在这个平台浏览好友的最新动态，并评论互动。随时随地分享自己的所感所悟，上传自己的精彩瞬间，随时查看。管理自己的详细信息，并可通过搜索查看陌生用户的信息，并发送添加好友请求来与陌生人建立好友关系，用户可管理自己的好友（删除、修改备注、移动分组等）和分组信息。

系统适用群体主要集中在80、90和00后青少年，当然其他年龄段的用户群体也存在，用户只要在有浏览器上输入地址正确登录后即可访问该系统，无需下载安装客户端。切本系统是基于网上已经存在的官方QQ系统发开而成，用户已经很熟悉本系统，用起来会非常顺手。

### 2.2.3 主要接口

用户接口：本系统的用户人群主要是青少年，故系统的人机交互就变得尤为重要。设计改变人类的生活，为人类创造了一个全新的具有物质文明和精神文明的社会。而科技的进步，技术的飞跃，人们对物质和精神需求的日益提高使得设计的发展空间越来越广。信息时代的来临，使人们的生活中充斥着各种各样的信息传播工具，于是人与机器的互动交流方式变得尤为重要，用户界面设计应运而生。而以视觉感知为主的图形用户界面成为如今最有效的信息传播媒介。良好的用户体验会吸引更多的用户群体来使用系统，所以用户界面的设计很大程度上决定了系统是否能很快的被用户接收及喜欢。而作为社交而生的QQ空间则是更大程度上加大了用户界面设计在系统总体设计开发中的占比，本系统由于引用了bootstrap流行框架，所以在界面设计上还是比较先进的，相信可以给用户带来比较良好的体验。

### 2.2.4 功能需求与非功能需求

系统功能需求包括：

用户注册登录后，进入个人中心，查看所有好友的最新动态，并可评论好友的动态。在我的好友菜单项中可以搜索陌生人的QQ账号来查看他的个人信息，并发送添加好友请求，对方验证请求信息后可执行同意或拒绝操作，同意后双方成为好友关系，拒绝后需要接收方暂时不再接受同一个用户的添加请求。用户可以管理自己的好友信息，包括修改好友备注，删除好友，屏蔽动态设置，访问我的空间权限，和移动到指定分组等功能。管理分组信息，包括重命名分组名，添加删除分组。好友信息管理和分组信息管理采用右键菜单弹出设置。用户可以发表说说、发表日志、给好友留言，上传精彩瞬间的照片，用于随时查看自己的记录。

系统非功能需求包括：

性能要求：系统能满足多人同时使用，页面反应时间不能超过6秒。

精度：为了查询完整，尽量让所有相应域包含查看关键字的记录都能查到。所以本系统的搜索好友采用模糊查询。

容错要求：为了保障系统能正常运行，尽量使得服务器数据库的数据合法化，以及前台能有效操作和浏览系统。

灵活性：搜索好友提供QQ账号搜索和QQ昵称搜索，为用户搜索好友提供了多种方式。当用户一定时间内处于未操作状态时，服务器不再响应对数据库的请求，即设置session过期时间，用户可通过再次登录来继续操作。

准确性和及时性：在系统设计和开发过程中，要充分考虑系统当前和将来可能承受的工作量，使系统的处理能力和响应时间能够满足用户对信息处理的需求。

数据的一致性与完整性：由于系统的数据时共享的，所以如何保证这些数据的一致性，是系统必须解决的问题，对于用户输入的数据，要为其定义完整性规则，如果不能符合完整性约束，系统应该拒绝该数据，数据合法性验证可以有效避免用户输入不合法的数据，同时利用JavaScript前台验证可以有效减缓项目后台及数据库操作的压力。

安全性分析：数据库采用Mybatis ORM映射进行数据操作，可避免SQL注入。

可更新性：对于系统可以根据用户需要进行更新，扩展其功能，并使系统稳定。

易用性和易维护性：由于本系统是基于已经存在的QQ系统做模仿实现，而QQ空间已经诞生十余年之久，用户群体已经非常熟悉系统操作，故本系统的操作在用户看来非常熟悉，很容易上手。易用性良好。同时在维护数据的时候应该更加的方便，避免因为误操作而导致数据的流逝。

系统运行要求：软件系统：Windows7及以上，IE浏览器8及以上。硬件要求：Intel Core i5-3230M CPU @2.6GHz，运行内存4G。

# 3 总体设计

## 3.1 系统框架设计

在前面的系统规划与需求分析的基础上，对该系统进行了总体设计。系统大致分为以下功能模块：用户注册登录；进入个人中心，查看所有好友的动态；评论说说，日志；给好友留言；搜索查看陌生人信息；发送添加好友请求；对方进行请求验证；管理好友信息，包括修改备注，删除好友，设置好友动态屏蔽，空间访问权限等。移动好友到指定分组；管理分组信息，包括重命名分组名称，删除分组，添加分组等。上传照片到我的相册。如下图3.1所示

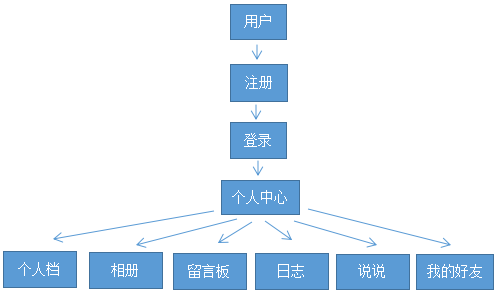


图3-1 系统框架

首先用户需要注册账号才能使用系统，注册功能验证通过后开始登录，登录成功首先进入个人中心页面，默认显示所有好友的动态并按最新时间进行排序显示，首次注册用户没有好友，需在我的好友菜单中进行搜索陌生人进行搜索并添加好友。用户可以通过导航栏菜单项进行功能切换，跳转到相应页面。

## 3.2 数据库设计

本系统采用MySQL数据库进行开发，并利用Power Design进行数据库表设计，结合Navicat Premium进行数据操作。共分为以下几张表：

用户信息表，如下3-1所示，该表主要存在用户的基本信息，包括用户账号和密码；个人详细信息，包括生日、性别、血型、情感状况（单身、恋爱中、已婚、未婚等）、个性签名、专业、公司、现居地址、家乡地址和星座等；空间设置，包括用户留言板主人寄语，最近登录系统的时间等。后期需求可以继续在此功能上添加。

表3-1 用户信息表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| userId | int | 用户Id |
| userAccount | varchar | 用户账号 |
| password | varchar | 用户密码 |
| userName | varchar | 用户昵称 |
| sex | varchar | 性别 |
| age | int | 年龄 |
| lastLoginTime | timestamp | 上次登录时间 |
| masterHope | varchar | 主人寄语 |
| personalSign | varchar | 个性签名 |
| bloodType | varchar | 血型 |
| emotion | varchar | 情感状况 |
| prefessional | varchar | 专业 |
| company | varchar | 公司 |
| hobby | varchar | 兴趣爱好 |
| currentAddress | varchar | 现居地址 |
| hometown | varchar | 家乡地址 |
| birthday | date | 生日 |
| constellation | varchar | 星座 |

好友关系表：存放好友map关系数据，用户添加好友，验证好友的数据都存放在这个表，同时这个表还存放用户对朋友的主要设置，包括：动态屏蔽，访问空间权限，特别关心状态，好友备注，所在分组等。一对好友通过两条数据来建立好友关系，并设置一个status字段来进行好友间添加请求，一方用户搜索并发送添加好友请求后，会在数据库生成两条数据，userId是用户Id，friendId是好友Id，但两条数据的状态初始为0（发送方）和4（接收方），然后接收方通过验证操作后，再修改这两条数据的status状态，当双方status都为1时，双方即可建立好友关系。 gNo字段用于管理好友所在分组，移动好友到指定分组时，只需要修改此字段为指定分组号即可。

表3.2 好友关系表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| friendmapId | int | 用户关系主键Id |
| userId | int | 用户Id |
| friendId | int | 好友Id |
| verifyInfo | varchar | 验证信息 |
| status | int | 好友状态：0.申请添加好友,等待验证；1.通过验证，添加成功；2.拒绝添加；3.删除；4.待验证 |
| addTime | timestamp | 添加时间 |
| beiZhu | varchar | 好友备注 |
| lastVisitTime | datetime | 上次访问时间 |
| gNo | varchar | 分组号 |
| allowVisited | int | 允许访问我的空间：0允许，1不允许 |
| hideDynamic | int | 隐藏他的动态：0：是，1：否 |
| careStatus | int | 特别关心：0是，1否 |

说说记录表，记录用户的说说，发表时间，内容等。分为物理主键说说Id、用户Id、发表说说时间、说说内容、及说说附带的图片。

表3.3 说说记录表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| talkId | int | 说说表主键Id |
| userId | int | 用户Id |
| talkTime | timestamp | 说说时间 |
| talkContent | text | 说说内容 |
| talkPic | varchar | 说说图片 |

留言信息表，用户查看好友给自己的留言，以及用户给好友发表留言等。messageId为物理主键Id，leaveId为发表留言的用户Id，masterId为被留言的用户Id。messageContent为留言内容，messageTime为发表留言时间。

表3.4 留言信息表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| messageId | int | 留言表主键Id |
| leaveId | int | 留言者Id |
| masterId | int | 主人Id |
| messageContent | varchar | 留言内容 |
| messageTime | timestamp | 留言时间 |

分组信息表：用户管理自己的分组信息，包括添加、删除、重命名指定分组。gId为分组表物理主键Id，userId为分组信息所属用户Id，gName为分组信息所属用户的分组名称，gNo为用户添加分组时根据固定规则生成的分组号，用于用户修改分组信息和添加好友到指定分组。分组表和用户关系表通过gNo关联，用户通过gNo来设置好友所在分组。

表3.5 分组信息表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| gId | int | 留言表主键Id |
| userId | int | 用户Id |
| gName | varchar | 分组名称 |
| gNo | varchar | 分组号，生成规则userId\_yyyyMMddHHmmss |

日志信息表，用户发表日志、查看日志的表。diaryId为物理主键Id，userId为发表日志的用户userId，diaryTime为发表日志时间。diaryContentId为int类型的日志Id，关联diarycontent表主键diaryContentId，在遍历查询用户的所有日志时查看日志信息表，显示日志标题和日志时间，然后在查看日志明细的时候，在根据日志内容主键Id即diaryContentId进行查询，查询出日志详细内容。

表3.6 日志信息表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| diaryId | int | 日志表主键Id |
| userId | int | 用户Id |
| diaryTime | timestamp | 日志时间 |
| diaryContentId | int | 日志内容表主键Id |
| diaryContentTitle | varchar | 日志标题 |

日志内容表：由于日志内容字段比较庞大，故采用varchar类型已不能满足需求，转而使用可以存储64kb的text类型进行日志内容存储，为了避免执行查询日志语句时给数据库带来压力过大而造成效率问题，故将日志内容单独拿出来独立一张表进行存储，并通过日志内容表主键diaryContentId和日志信息表进行关联。

表3.7 日志内容表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| diaryContentId | int | 日志内容表主键Id |
| diaryContentText | text | 日志内容 |
| diaryContentTitle | varchar | 日志标题 |
| diaryContentTime | timestamp | 发表日志时间 |

评论表：关联用户说说表和日志表，通过type字段来区分好友评论的说说和日志。并通过cUserId和aUserId来关联评论者和被评论者的用户表主键userId。

表3.8 评论表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| commentId | int | 评论表主键Id |
| cUserId | int | 评论者Id |
| aUserId | int | 被评论者Id |
| commentType | int | 评论类型：0：说说，1：日志 |
| commentContent | varchar | 评论内容 |
| commentTime | timestamp | 评论时间 |
| commentTypeId | int | 评论类型的主键Id：说说：说说表主键Id，日志：日志表主键Id |

照片表，用户上传的照片地址信息表，该表存放照片主键Id，由用户Id+下划线+上传时间拼接而成生成唯一主键。每个用户上传的照片全部单独存放在本机指定上传文件夹下的单独目录，及在上传照片的时候为每个用户分配的一个单独文件夹存放用户的照片，改文件夹由用户账号命名，如果不存在该目录，则在上传的时候动态创建一个该文件夹。这样做有利于照片在本地的查看。所在相册Id，照片存放在本机路径地址，上传时间以及照片的名字等。

上传照片查看时根据tomcat配置web module 设置指定目录，然后拼接数据库的存放地址来指定本机照片地址全路径。

表3.9 照片表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| photoId | int | 照片表主键Id |
| userId | int | 用户Id |
| photoDirId | int | 照片所在相册Id |
| photoUrl | varchar | 照片存放地址 |
| uploadTime | timestamp | 照片上传时间 |
| photoName | varchar | 照片名字 |

相册表：存放用户的所有相册，用户上传的所有照片都有一个字段为相册表主键Id，用来区分照片所属相册，进行相册之间的移动，也是直接修改照片信息表的该字段。

表3.10 相册表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 描述 |
| photoDirId | int | 相册表主键Id |
| userId | int | 相册所属用户ID |
| photoDirName | varchar | 相册名字 |
| createTime | datetime | 创建时间 |

## 3.3 系统主要模块设计

本系统共分为七个模块，分别为个人中心、个人档，相册，留言板，日志，说说，我的好友。用户首先通过注册来创建一个账号，成功登陆后进入个人中心，在导航栏可以通过点击不同模块的菜单项来进行功能切换。

### 3.3.1 个人档

该模块是用户管理个人信息的页面，用户可以在这个页面查看编辑自己的个人信息。

### 3.3.2 个人中心

用户登录之后默认跳转的页面，该模块显示用户所有好友的动态并按时间最新排序。用户可以在该模块浏览到好友的最新动态并可进行评论互动。个人中心也可以发表说说，并同步动态到所有好友动态中。

### 3.3.3 相册

管理用户照片的模块。用户可以上传自己的精彩瞬间，并保留在空间的相册下，随时查看。在这里你可以随时随地地收藏分享喜欢的图片，在分享中发现更多志趣相投的朋友。可以对相册进行管理，包括添加删除相册、重命名相册。也可以对照片进行管理，包括批量移动到指定相册、删除照片。

### 3.3.4 留言板

管理用户来自好友的留言信息。当用户不在线上时，好友可以在这里留下信息，用户可以随时查看自己的留言板来查看好友留给他的信息。

### 3.3.5 日志

管理用户的日志，通过写日志的方式来记录生活的点点滴滴，用户可以像写博客一样在这里发表仅供好友能查看的文章。

### 3.3.6 说说

管理用户的说说，用一条不超过二百个字的说说来发表用户的所见所闻所感。并实时发表说说同步到好友动态中，好友可以即时查看到彼此的最新动态并参与评论互动，对彼此的说说发表看法。

### 3.3.7 好友

管理用户的好友，包括搜索添加好友、设置好友权限、分组信息等。也可实时查看来自陌生人的好友添加请求。

# 4 详细设计和实现

## 4.1 系统主要程序设计与实现

系统中涉及的主要功能分别为好友管理、说说、相册、留言板、日志。以下分别就这几个模块中的核心功能分析设计与实现的思路。

### 4.1.1 好友

QQ空间作为社交平台，用户可以在这里分享动态，也可以查看别人的动态，好友功能必不可少，好友管理就成为重中之重。

用户成功登录后，在导航栏点击“好友”菜单，即进入好友页面。

在好友页面，有三个子菜单，分别为：好友列表、寻找好友、好友请求。

好友列表-左侧好友分页展示：

默认按照最近访问时间进行排序，分页显示好友。在该菜单下面，可以对好友进行一些简单设置，设置好友特别关心/取消特别关心，允许访问/禁止访问，显示动态/隐藏动态。相关代码如下所示：

JavaScript代码

function updateFriendRight(param1,param2,type,path){

$.ajax({

url:path+"/friend/updateFriendRight.action",

type: "POST",

data:{friendId:param1,option:param2,type:type},

cache: false,

dataType:"json",

success: function(data) {

if(data == '0'){

window.location.href=path+"/friend/myfriends.action";

}else if(data=='1'){

alert("数据库修改失败！");

}else if(data=='-1'){

alert("session失效，请重新登录");

}else{

alert("操作失败，其他未知错误。");

}

}

});

}

jsp页面修改权限的三个功能按钮都调用同一个JavaScript的ajax方法，根据传入参数type来判断权限类型，进而让controller来执行不同的业务操作，并动态刷新页面。页面采用bootstrap框架来进行界面UI设计。

controller方法

/\*\*

\* 修改指定用户的权限：访问空间，动态屏蔽

\*/

@RequestMapping("/updateFriendRight")

public void updateFriendRight(Model model,HttpServletRequest request,HttpServletResponse response){

String friendIdString=request.getParameter("friendId");

String optionString=request.getParameter("option"); //操作状态

String type=request.getParameter("type"); //操作类别：allowVisited、hideDynamic

int friendId=-1;

int option=-1;

int userId=-1;

if(friendIdString!=null && optionString!=null){

friendId=Integer.parseInt(friendIdString);

option=Integer.parseInt(optionString);

}

User user=(User) request.getSession().getAttribute("user");

int updateRow=0;

String result="";

if(user!=null){

userId=user.getUserId();

Map<String, Integer> map=new HashMap<>();

map.put("userId",userId);

map.put("friendId",friendId);

//根据不同类别封装不同字段

if("allowVisited".equals(type)){

map.put("allowvVisited",option);

//修改访问空间权限

updateRow=friendmapService.updateFriendRight(map);

}else if ("hideDynamic".equals(type)) {

map.put("hideDynamic",option);

//修改动态屏蔽状态

updateRow=friendmapService.updateFriendDynamic(map);

}else if ("careStatus".equals(type)) {

map.put("careStatus",option);

//修改特别关心状态

updateRow=friendmapService.updateCareStatus(map);

}

if(updateRow==1){

result="0";

}else if(updateRow==0){

result="1";

}

}else{

result="-1";

}

log.info("ajax修改访问权限，用户：{},好友：{},修改类别：{},修改状态：{},result：{}", user.getUserName(),friendId,type,option,result);

response.setCharacterEncoding("utf-8");

response.setContentType("application/json;charset=utf-8");

try {

response.getWriter().write(result);

} catch (IOException e) {

e.printStackTrace();

}

}

controller中进行type参数判断，来区分需要修改的权限，进而执行不同的数据库更新操作。该三个权限在数据库中利用friendmap表中的三个int字段（allowVisited、hideDynamic、careStatus）来进行标识，并统一约定：0是、1否。

好友列表-右侧好友分组：

好友分组实现好友列表展示和收缩功能，jsp页面遍历分组集合和好友集合，利用JavaScript进行列表动态展示。本功能采用分组和好友两个表来完成，在好友表friendmap中存放着每个用户的好友信息，每个好友信息都有一个分组字段来标识他所在的分组。然后就可以利用好友的分组字段来遍历，分组显示在对应的分组列表下面。

好友分组还支持右键菜单功能，在好友列表上鼠标右键执行点击操作，即弹出对好友的基本操作：修改备注、移动到分组、删除好友等操作；在分组信息列表上鼠标右键执行点击操作,即弹出对该分组的基本操作：删除分组、添加分组、重命名分组。

右键菜单使用smartMenu插件，结合自定义JavaScript鼠标事件监听来实现，核心JavaScript代码如下：

$(".friendsGroup").bind("mousedown",(function(e){

if(e.which == 3){

//js右键触发事件

// console.log("所选内容为:cellIndex: "+e.target.cellIndex+",InnerHtml:"+e.target.innerHTML+",nodeName:"+e.target.nodeName); var node1=e.target;

var hidden= $(node1).next("input").val();

//自定义右键上下文

var objUpdateBeiZhu = {

text: "修改备注 ",

func: function() {

/\* funTrGet().remove(); \*/

}

},

objGotoZone = {

text: "进入QQ空间",

func: function() {

/\* funTrGet().hide(); \*/

}

},

objDeleteFriend= {

text: "删除好友",

func: function() {

/\* funTrGet().hide(); \*/

}

},

objDeleteGroup = {

text: "删除分组",

func: function() {

/\* funTrGet().css("font-weight", "700"); \*/

}

},

objAddGroup = {

text: "添加分组",

func: function() {

/\* funTrGet().css("font-weight", "300"); \*/

}

},

objRenameGroup={

text: "重命名分组",

func: function() {

/\* funTrGet().css("font-weight", "500"); \*/

}

},

objMoveToGroup = {

text: "移动到分组",

func: function() {

$("#moveToGroupFriendId").val(hidden);

$("#moveToGroupButton").click(); $("#moveToGroupFriendName").html(e.target.innerHTML);

}

};

var groupsMenuData = [

[objDeleteGroup,objAddGroup,objRenameGroup]

];

$(".RightKeyOfGroups").smartMenu(groupsMenuData, {

name: "groups",

beforeShow: function() {

$.smartMenu.remove();

}

});

var friendsMenuData = [ [objUpdateBeiZhu,objMoveToGroup,objDeleteFriend],[objGotoZone]

];

$(".RightKeyOfFriends").smartMenu(friendsMenuData, {

name: "friends",

beforeShow: function() {

$.smartMenu.remove();

}

});

}

}));

其中，移动好友至分组采用弹出模态框设计，在模态框中遍历分组信息，然后选择指定分组点击保存后，好友及被移动到指定分组下面。该功能利用JavaScript监听右键点击事件，当鼠标在指定区域点击右键后（JavaScript中e.which == 3）然后smartMenu在右键点击时抓取好友Id，再结合模态框的分组id一起通过ajax传给后台controller进行好友分组信息更新。

寻找好友-左侧查找添加好友：

查找好友采用模糊查询，分为QQ昵称查找和QQ账号查找

jsp页面采用JavaScript正则验证，对利用QQ账号查找的input框进行正则验证，只允许用户输入9位纯数字，而QQ昵称则只做长度验证，因为QQ昵称是随意的，用户既有可能纯数字也有可能纯汉字或者数字汉字结合。

对遍历出来的所有查询结果，列表显示每个用户的昵称和账号信息。用户可以点击查询结果每条数据后面的查看按钮来查看搜索结果中指定用户的详细信息。点击添加按钮，弹出验证信息模态框，输入完成 后发送给指定用户添加好友请求。

核心代码：

/\*\*

\* 添加好友

\*/

@RequestMapping("/addFriend")

@ResponseBody

public String addFriend(HttpServletRequest request){

User user=(User)request.getSession().getAttribute("user");

String friendIdString =request.getParameter("friendId");

int friendId=0;

if(friendIdString!=null){

friendId=Integer.parseInt(friendIdString);

}

int userId=0;

if(user!=null){

userId=user.getUserId();

}

log.info("userId:{},friendId:{}",userId,friendIdString);

Map<String, Integer> map=new HashMap<>();

map.put("userId", userId);

map.put("friendId", friendId);

//添加之前先判断是否已经是好友状态，待对方验证，已加为好友,对方已经向你发送好友请求（此时直接更新双方status为1，及双方成为好友）

Friendmap verifyfriend=null;

verifyfriend=friendmapService.findVerifyfriend(map);

log.info("verifyFriend:{}",verifyfriend);

if(verifyfriend!=null){

if(verifyfriend.getStatus()==1){

//已加为好友，不可在此添加

return "1";

}else if(verifyfriend.getStatus()==0){

//验证中，不可重复添加

return "0";

}else if(verifyfriend.getStatus()==2){

//已拒绝，重新添加

return "2";

}else if(verifyfriend.getStatus()==4){

//对方已经向你发送好友添加请求，此时再次添加直接修改双方status 为1，及成为好友

Friendmap fm1=new Friendmap();

Friendmap fm2=new Friendmap();

int num1=-1;

int num2=-1;

fm1=new Friendmap();

fm1.setUserId(userId);

fm1.setFriendId(friendId);

fm1.setStatus(1);

num1=friendmapService.updateFriendRequest(fm1);

fm2=new Friendmap();

fm2.setUserId(friendId);

fm2.setFriendId(userId);

fm2.setStatus(1);

num2=friendmapService.updateFriendRequest(fm2);

log.info("【查找好友】中点击【添加】结果：userId:{},friendId:{},num1:{},num2:{}",userId,friendId,num1,num2);

if(num1 > 0 && num2 > 0){

return "4";

}

}

}

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

//String beizhu=request.getParameter("beizhu");

log.info("verifyInfo:{}");

//若查找出的verifyFriend为空，则未添加过该好友，可以插入数据

Friendmap fm1=new Friendmap();

fm1.setUserId(userId);

fm1.setFriendId(friendId);

fm1.setVerifyInfo(verifyInfo);

fm1.setStatus(0);

//fm1.setBeiZhu(beizhu);

//申请方插入friendmap数据

int num1=friendmapService.addFriend(fm1);

Friendmap fm2=new Friendmap();

fm2.setUserId(friendId);

fm2.setFriendId(userId);

//fm2.setVerifyInfo("");

fm2.setStatus(4);

//fm2.setBeiZhu("");

//接收方插入friendmap数据

int num2=friendmapService.addFriend(fm2);

if(num1>0 && num2>0){

return "success";

}else{

return "false";

}

}

好友请求:

用户可以通过切换好友导航栏下面的三个子菜单按钮来转换至好友请求菜单项，该菜单项下面显示所有来自其他用户或好友的请求，用户可以通过点击查看按钮来查看来自其他用户的详细信息，然后可以点击“同意”来添加用户为好友，或点击“拒绝”来拒绝添加用户，被拒绝的用户可以再次发送添加好友请求。

处理好友添加请求核心代码：

/\*\*

\* 处理好友添加请求 0：同意,1：拒绝

\*/

@RequestMapping("/updateFriendRequest")

@ResponseBody

public String updateFriendRequest(HttpServletRequest request){

User user=(User) request.getSession().getAttribute("user");

int userId=-1;

if(user!=null){

userId=user.getUserId();

String friendIdString=request.getParameter("friendId");

String typeString=request.getParameter("type");

log.info("user:{},FriendIdString:{},typeString:{}",user,friendIdString,typeString);

int friendId=-1;

int type=-1;

int num1=0;

int num2=0;

Friendmap fm1=null;

Friendmap fm2=null;

if(friendIdString!=null && typeString!=null){

friendId=Integer.parseInt(friendIdString);

type=Integer.parseInt(typeString);

if(type==0){ //同意添加，修改两条数据：userId的status由4置1，friendId的status由0置1

fm1=new Friendmap();

fm1.setUserId(userId);

fm1.setFriendId(friendId);

fm1.setStatus(1);

num1=friendmapService.updateFriendRequest(fm1);

fm2=new Friendmap();

fm2.setUserId(friendId);

fm2.setFriendId(userId);

fm2.setStatus(1);

num2=friendmapService.updateFriendRequest(fm2);

log.info("【同意】添加结果：userId:{},friendId:{},type:{},num1:{},num2:{}",userId,friendId,type,num1,num2);

if(num1 > 0 && num2 > 0){

return "0";

}

}else if(type==1){ //拒绝添加,修改一条数据，userId的status由4置2

fm1=new Friendmap();

fm1.setUserId(userId);

fm1.setFriendId(friendId);

fm1.setStatus(2);

num1=friendmapService.updateFriendRequest(fm1);

log.info("【拒绝】添加结果：userId:{},friendId:{},type:{},num1:{}",userId,friendId,type,num1);

if(num1 > 0 ){

return "1";

}

}

}

return "-2";

}else{

return "-1";

}

}

### 4.1.2 说说

说说功能是QQ空间的核心功能，用户注册QQ空间的初衷就是可以利用这个平台分享自己的所见所闻所感，同时，用户还可以即时查看到所有好友的最新动态，在了解好友的生活动态的基础上，来进行互动评论，让远在世界各地的亲朋好友不再拥有距离感。

用户在成功登陆之后，即可进入个人中心页面，用户所有好友的说说动态按时间倒序（即最新时间）展示。用户可以在这个页面直接发表说说，并利用ajax动态加载，即时显示在所有动态中。也可以在菜单项说说中，直接查看用户自己的说说，该菜单项下边只会显示用户自己发表过的说说内容，并按最新时间倒序显示。用户也可以在这里发表说说，及时显示在我的所有说说列表中，并同步说说动态到所有好友动态中。该页面用户自己的说说采用分页显示，用户可以点击上一页下一页来进行查看更多说说记录。

所有好友的说说动态分段显示，即每次只为用户显示指定数量的好友动态，在用户点击查看更多的时候，利用Ajax异步方法加载数据并动态添加到页面上。

用户可对好友的动态进行评论，评论内容单独占用一个评论表，该评论表关联发表评论的用户Id和被评论者用户Id，以及评论类型和评论类型表主键Id，评论类型可以为说说、日志等，然后评论类型表主键Id及管理被评论者发表的说说或者日志。及当用户评论另一个用户的说说时，该评论内容包括两个用户的Id和关联的说说Id，该条评论被插入评论表中。

查看评论的时候，点击说说下面的评论按钮，即可调用ajax方法查看评论列表，并动态追加到说说下面显示出来。

### 4.1.3 相册

在菜单项里选择相册功能，即可进入我的相册页面，该页面保存着用户上传的照片。

用户通过导航栏里的相册标签，可以进入个人相册展示页面。该页面默认显示该用户所有的相册列表，用户可以在相册列表上方选择相册管理选项，包括创建相册、上传照片到指定相册

每个相册可以单独设置，包括修改相册名字和删除相册。

点击单个相册，可以进入该相册，查看该相册下的所有照片。也可以在该相册下批量上传照片。同时支持对所有的照片的管理，包括：批量移动到相册、删除照片。

用户在访问好友的空间的时候，值拥有访问相册的权限，所有对相册和照片的管理操作被屏蔽掉。

### 4.1.4 留言板

在菜单项里选择留言板功能，即进入我的留言板页面。显示所有好友给用户的留言，按最新留言时间分页显示，用户可以设置自己的主人寄语。

用户只可以查看好友给自己的留言，不能给自己发表留言，可以访问好友的空间后，在好友的留言板给好友留言。

### 4.1.5 日志

日志功能提供给用户发表日记功能，用户可以随时随地编辑保存自己的文章，并可发表到朋友圈，其他用户可以实时查看该用户发表的日志动态，并可查看原文并发表评论。

日志列表默认展示日志标题、发表时间按发表时间最新排序分页显示，用户自己可以通过点击日志标题链接来查看编辑日志原文，日志内容和日志原文采用两个表来存储。主要是为了减轻数据库的操作压力，因为日志文本字数比较多，故设计为text类型，然后独自存放在另外一张表中，两张表通过日志id来关联查询。

写日志的时候可以通过下拉框选项选择日志背景，这样设计极大的提升了人机交互性。发表后的日志，用户也可以进行修改和删除操作。

好友可以通过日志标题来链接进入日志正文浏览日志，并可进行评论。

# 5 系统测试与维护

## 5.1 软件测试

软件开发是一项集信息量大、程序代码多和时间长的工程，一个软件的好坏、质量的优劣不仅取决于软件的开发目标及其可行性的评估、功能需求的分析、软件的设计以及程序代码的编写，还取决于软件测试。软件测试是软件开发中的最后一个阶段，软件测试是使用人工或者自动手段来运行或测试某个系统的过程，通过测试发现软件开发设计过程中存在的问题，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。

现针对已设计的核心模块，拟进行以下几项测试用例，测试系统核心模块能否正常使用。

### 5.1.1 登录验证测试

用户登录时进行账号验证，该验证包括输入合法性验证和账号密码正确性验证测试。

1.账号密码输入合法性验证测试：

1. 输入：非纯数字账号

期望输出：提示用户账号不能包括数字以外的字符，输入不正确，请重新输 入。

测试结果：tip小提示红色字体显示提示信息，用户名为纯数字组合。

1. 输入：账号位数多于9位或者少于9位

期望输出：提示账号为9位纯数字组成

测试结果：tip小提示红色字体显示提示信息，输入账号非法，用户名长度为 9位。

2.账号密码正确性验证：

（1）输入：错误的账号和密码

期望输出：提示用户账号或密码错误，请重新输入

测试结果：tip小实体红色字体显示提示信息，账号或密码错误，请重新输入 账号密码

（2）输入：账号或者密码为空

期望输出：账号和密码为必填项

测试结果：账号和密码为必填项

测试截图：



### 5.1.2 注册功能测试

用户在登录页面点击注册账号即可进入注册账号页面：

1.注册信息非空验证：

（1）输入：所有必填项中，有一个没有输入框为空，没有填写信息

期望输出：提示用户所有注册项均不能为空

测试结果：提示用户所有注册项均不能为空

2.注册信息正确性验证：

（1）输入：7位账号

期望输出：账号为9位纯数字组合，请重新输入

测试结果：账号为9位纯数字组合，请重新输入

（3）输入：账号包含非数字字符

期望输出：账号为9位纯数字组合，请重新输入

测试结果：账号为9位纯数字组合，请重新输入

测试截图：



### 5.1.3 好友功能测试

1. 模糊查询好友

（1）输入：好友账号或昵称

期望输出：所有包含账号或昵称的QQ好友列表

测试结果：界面显示所有包含账号或昵称的QQ好友列表

2.查看搜索结果中的好友详细信息

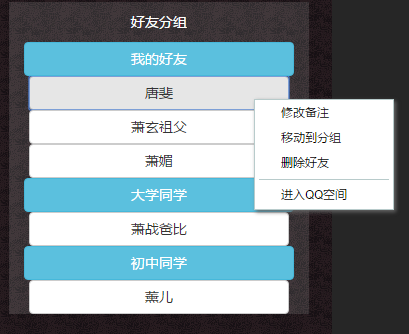
（1）输入：在搜索结果中任选一个QQ用户点击查看

期望输出：界面显示该用户的详细信息

测试结果：右侧列表显示出该用户的详细信息

测试截图：





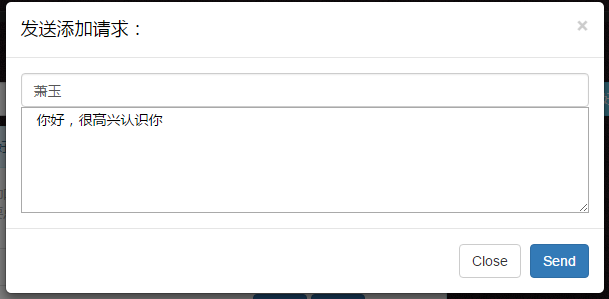
1. 添加好友测试

（1）输入：点击搜索结果中任一用户的添加按钮，发送添加好友请求

期望输出：发送请求成功，目标用户收到添加好友请求

测试结果：提示发送请求成功，等待对方确认验证信息

测试截图：



1. 确认添加请求

（1）输入：用户收到添加好友请求，点击查看显示发送请求的用户信息

期望输出：界面显示发送请求的用户信息

测试结果：右侧模态框显示用户的详细信息

（2）输入：用户点击同意或拒绝按钮

期望输出：同意点击后，两个用户成为好友，点击拒绝后，请求界面显示 已拒绝。

测试结果：同意点击后，两个用户成为好友，点击拒绝后，请求界面显示 已拒绝。

（3）如果用户先前已经收到来自好友的添加好友，那么自己再次发送给对方添加 请求后，双方立即成为好友，不需经过同意操作。

输入：在对方已经发送给自己添加请求后，自己也给对方发送添加请求

期望输出：弹框提醒，恭喜你们成为好友

测试结果：弹框提醒，恭喜你们成为好友

测试截图：



1. 移动好友到指定分组

（1）输入：在好友分组界面，选择一个好友右键移动到分组，出现分组名称，选 中一个点击

期望输出：移动好友到分组成功

测试结果：移动好友到分组成功

1. 移动好友到指定分组

（1）输入：在好友分组界面，选择一个好友右键移动到分组，出现分组名称，选 中一个点击

期望输出：移动好友到分组成功

测试结果：移动好友到分组成功

测试截图：



1. 访问好友空间

（1）输入：在好友分组界面，选择一个好友右键进入QQ空间

期望输出：顶部导航栏变为 他的主页，上次访问时间，返回个人中心，以及 进入该好友说说列表页面。

测试结果：顶部导航栏变为 他的主页，上次访问时间，返回个人中心，以及 进入该好友说说列表页面。

### 5.1.4 说说发表测试

由于在评论好友的说说时，分为在自己的空间评论好友的说说和在好友的空间评论好友的说说，这两种情况下，session域中存储用户key值不同，而评论系统设计根据session域用户key动态显示权限和评论者名字，及其他好友的评论显示好友昵称和回复该条评论功能，而自己的评论显示昵称为“我”及删除该条评论功能。故在这里需要测试，在不同的用户空间背景下，评论系统能否正常执行

1. 在个人中心或我的说说界面,发表说说，动态即时显示出来。

（1）输入：文字或表情，点击发表

期望输出：发表成功，动态被同步到所有好友动态中

测试结果：发表成功，动态被同步到所有好友动态中

1. 在自己空间的个人中心中，评论好友的说说

（1）输入：点击好友说说下面的评论按钮，添加文字或表情，点击发表

期望输出：评论成功，评论内容动态显示在页面上，评论者显示“我”

测试结果：评论成功，评论内容动态显示在页面上，评论者显示“我”

3.在好友空间的说说列表中，评论好友的说说

（1）输入：点击好友说说下面的评论按钮，添加文字或表情，点击发表

期望输出：评论成功，评论内容动态显示在页面上，评论者显示“我”

测试结果：评论成功，评论内容动态显示在页面上，评论者显示“我”

测试截图：



### 5.1.5 相册上传照片测试

（1）输入：点击上传按钮，选择非图片文件

期望输出：系统提示上传文件格式错误

测试结果：系统弹框提示，选择的文件不是图片，请重新选择

（2）输入：点击上传按钮，选择多张图片

期望输出：选定之后，上传之前，页面显示要上传图片的缩略图，上传之后， 图片显示在页面中。

测试结果：选定之后，上传之前，页面显示要上传图片的缩略图，上传之后， 图片显示在页面中。

（3）输入：选取多张图片，批量移动到指定相册

期望输出：操作完成，照片被移动到目标相册

测试结果：操作完成，照片被移动到目标相册

（4）输入：选择指定相册，点击删除相册

期望输出：提示该相册下所有照片均会被删除，请确认操作，确认后，该相 册及相册下所有照片均被删除

测试结果：提示该相册下所有照片均会被删除，请确认操作，确认后，该相 册及相册下所有照片均被删除

测试界面：



### 5.1.6 留言板发表测试

（1）输入：输入留言信息，点击留言给好友留言

期望输出：留言信息显示在好友留言板中

测试结果：留言信息显示在好友留言板中

测试截图：

### 5.1.7 日志测试

（1）输入：点击写日志，输入日志文字，点击发表

期望输出：日志成功发表，并显示在日志列表中。

测试结果：日志成功发表，并显示在日志列表中。

（2）输入：点击修改日志，修改日志内容和日志背景，点击保存

期望输出：修改成功

测试结果：修改成功

（3）输入：进入好友空间，选择日志列表，给其中一篇日志发表评论

期望输出：评论成功

测试结果：评论成功

（4）输入：进入自己的日志列表，选择一篇日志点击删除

期望输出：删除成功

测试结果：删除成功

# 结论

本系统采用SSM框架进行设计，对我来说是一次新的突破，SSM框架指的是SpringMVC 做控制器(controller)，Spring 管理各层的组件，MyBatis 负责持久化层。相比于Struts2的Action类级别，SpringMVC的方法级别，更容易实现RESTful风格。搭配上log4j和slf4j的日志系统，日志打印更加便捷。

经过本次的毕业设计，我发现书本上的理论知识真的是不够用，只有平时积攒下的代码行数及bug经验教训才能让你的项目接近完美。相比当初刚接触java时候的自己，对javaWeb的开发确实是进步了很多，很多当初不会的技术，现在也慢慢掌握了。

即便如此在做毕设的过程中，我也遇到过不少技术难题，也有很多次查找百度解决难题，以及寻求同学的帮助。在这个过程中，我又一点点提升了技术。实践是锤炼技术的唯一途径。

虽然项目做的还不完善，但对于QQ空间的基本运作，我也是更加清晰了很多，今后还要继续加油，不断的学习新的技术，不断的充实自我，让自己的技术更进一步的提升。

# 参考文献

[1]李雪莲,办公自动化系统分析与开发[D],吉林:吉林大学,2010

[2]刘凯立,SSH技术研究及其在办公自动化系统中的应用[D],北京:中国地质大学,2010

[3]欧海华,安全高效的中小企业办公自动化系统设计[D],重庆:重庆大学,2008

[4]朱雪琴,常建功,Java Web项目开发案例导航[M],北京:电子工业出版社,2012

[5]杨春花,Spring攻略[M],人民邮电出版社,2009

[6]孙更新等,Java程序开发大全[M],中国铁道出版社,2010

[7]杨少波,顾益军,J2EE项目实训Spring框架技术[M],清华大学出版社,2008

[8]米奕杭,基于SSH框架的网络购物电子商务平台的设计与实现[D],四川:四川师范大学,2010

[9]夏昕,深入浅出Hibernate[M],电子工业出版社,2010

[10]刁成嘉,UML系统建模与分析设计[M],北京:机械工业出版社,2007

[11]易琴,重庆职业学院办公自动化系统的设计与实现[D],重庆:重庆职业学院,2008

[12]Seth Ladd,Expert Spring MVC and Web Flow[M],人民邮电出版社,2008

[13]Paul Deck,Spring MVC学习指南[M],人名邮电出版社，2015

[14]Praveen.Gupta,Spring Web MVC Framework for rapid open source J2EE application development: a case study[J],International Journal of Engineering Science and Technology, 2010

# 致谢

在整个毕设的开发过程中，我遇到了不少的技术难题以及bug，在和同学好友的讨论下，一个个bug和技术难题都变得迎刃而解。我也积累了不少的经验。在和同学们一起奋斗的过程中，我感到非常的充实与快乐。

感谢指导老师的悉心指导，替我理清了开发的思路。学海无边，要掌握的技术还远远不够，只有不断的向别人学习，才能不断的进步。

# 外文原文

Abstract: Enterprise-class solution, the balance is from the "heavyweight" system, such as EJB, to lightweight frameworks such as Spring tilt. Spring offers a wealth of practical service, it put a lot of competitive open source projects into a single framework. In many cases, Spring framework for J2EE systems are replacing traditional, more and more developers want to get some experience on the Spring skills.

Key words: j2ee; component; Distributed; Dynamic Deployment;springMVC

1. Introduction: Spring the Application Framework

When first confronted with the Spring Framework, one might be tempted to think: "Oh no, not yet another web framework". This article will outline why Spring isn't particularly a web framework but a generic lightweight application framework with dedicated web support, and show the architectural differences to Struts and WebWork

In contrast to Struts or WebWork, Spring is an application framework for all layers: It offers a bean configuration foundation, AOP support, a JDBC abstraction framework, abstract transaction support, etc. It is a very non-intrusive effort: Your application classes do not need to depend on any Spring classes if not necessary, and you can reuse every part on its own if you like to. From its very design, the framework encourages clean separation of tiers, most importantly web tier and business logic: e.g. the validation framework does not depend on web controllers. Major goals are reusability and testability: Unnecessary container or framework dependencies can be considered avoidable evils.

Of course, Spring's own web support is nicely integrated with the framework's general patterns. Nevertheless, replacing the web solution with Struts, WebWork, or the like is easy. Both with Spring's web support or a different one, Spring allows for building a true dedicated middle tier in the web container, with the option to reuse exactly the same business logic in test environments or standalone applications. And within J2EE, your business logic will not unnecessarily depend on container services like JTA or EJB - allowing complex, well-architected web applications to run in a "simple" container like Tomcat or Resin.

Note that Spring doesn't generally aim to compete with existing solutions. It rather fosters seamless integration with standards like Servlet, JSP, JTA, JNDI, JDBC, and JDO, and well-suited tools like Hibernate, Velocity, Log4J, and Caucho's Hessian/Burlap. The framework is designed to grow with the needs of your applications, in terms of technology choice: For example, you will probably use JTA via Spring's JtaTransactionManager if you need distributed transactions - but only then, as there are perfect replacements for single databases, like DataSourceTransactionManager or HibernateTransactionManager.

2.      Web MVC: The Design of Spring's Web Framework

Spring's web framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings, view resolution, and locale and theme resolution. The default handler is a very simple Controller interface, just offering a "ModelAndView handleRequest(request,response)" method. This can already be used for application controllers, but you will prefer the included implementation hierarchy, consisting of AbstractController, AbstractCommandController, MultiActionController, SimpleFormController, AbstractWizardFormController. Application controllers will typically be subclasses of those. Note that you can choose an appropriate base class: If you don't have a form, you don't need a FormController. This is a major difference to Struts.

You can take any object as command or form object: There's no need to implement an interface or derive from a base class. Spring's data binding is highly flexible, e.g. it treats type mismatches as validation errors that can be evaluated by the application, not as system errors. So you don't need to duplicate your business objects' properties as Strings in your form objects, just to be able to handle invalid submissions, or to convert the Strings properly. Instead, it's often preferable to bind directly to your business objects. This is another major difference to Struts which is built around required base classes like Action and ActionForm - for every type of action.  Compared to WebWork, Spring has more differentiated object roles: It supports the notion of a Controller, an optional command or form object, and a model that gets passed to the view. The model will normally include the command or form object but also arbitrary reference data. Instead, a WebWork Action combines all those roles into one single object. WebWork does allow you to use existing business objects as part of your form, but just by making them bean properties of the respective Action class. Finally, the same Action instance that handles the request gets used for evaluation and form population in the view. Thus, reference data needs to be modelled as bean properties of the Action too. These are arguably too many roles in one object.

Regarding views: Spring's view resolution is extremely flexible. A Controller implementation can even write a view directly to the response, returning null as ModelAndView. In the normal case, a ModelAndView instance consists of a view name and a model Map, containing bean names and corresponding objects (like a command or form, reference data, etc). View name resolution is highly configurable, either via bean names, via a properties file, or via your own ViewResolver implementation. The abstract model Map allows for complete abstraction of the view technology, without any hassle: Be it JSP, Velocity, or anything else - every renderer can be integrated directly. The model Map simply gets transformed into an appropriate format, like JSP request attributes or a Velocity template model.

3.      Integration: Using a Different Web Framework with Spring

Many teams will try to leverage their investments in terms of know-how and tools, both for existing projects and for new ones. Concretely, there are not only a large number of books and tools for Struts but also a lot of developers that have experience with it. Thus, if you can live with Struts' architectural flaws, it can still be a viable choice for the web layer. The same applies to WebWork and other web frameworks.

If you don't want to use Spring's web MVC but intend to leverage other solutions that Spring offers, you can integrate the web framework of your choice with Spring easily. Simply start up a Spring root application context via its ContextLoaderListener, and access it via its ServletContext attribute (or Spring's respective helper method) from within a Struts or WebWork action. Note that there aren't any "plugins" involved, therefore no dedicated integration: From the view of the web layer, you'll simply use Spring as a library, with the root application context instance as entry point. All your registered beans and all of Spring's services can be at your fingertips even without Spring's web MVC. Spring doesn't compete with Struts or WebWork in this usage, it just addresses the many areas that the pure web frameworks don't, from bean configuration to data access and transaction handling. So you are able to enrich your application with a Spring middle tier and/or data access tier, even if you just want to use e.g. the transaction abstraction with JDBC or Hibernate.

4.      Feature Check List

If just focussing on the web support, some of Spring's unique features are: .Clear separation of roles: controller vs validator vs command object vs form object vs model object, DispatcherServlet vs handler mapping vs view resolver, etc. .Powerful and straightforward configuration of both framework and application classes as JavaBeans, including easy in-between referencing via an application context, e.g. from web controllers to business objects and validators. .Adaptability, non-intrusiveness: Use whatever Controller subclass you need (plain, command, form, wizard, multi action, or a custom one) for a given scenario instead of deriving from Action/ActionForm for everything. .Reusable business code, no need for duplication: You can use existing business objects as command or form objects instead of mirroring them in special ActionForm subclasses. .Customizable binding and validation: type mismatches as application-level validation errors that keep the offending value, localized date and number binding, etc instead of String-only form objects with manual parsing and conversion to business objects. Customizable handler mapping, customizable view resolution: flexible model transfer via name/value Map, handler mapping and view resolution strategies from simple to sophisticated instead of one single way. Customizable locale and theme resolution, support for JSPs with and without Spring tag library, support for JSTL, support for Velocity without the need for extra bridges, etc. Simple but powerful tag library that avoids HTML generation at any cost, allowing for maximum flexibility in terms of markup code.

5.      The Applications of Spring Framework in Web

Spring contains a lot of functionality and features, which are well-organized in seven modules shown in the diagram below. This section discusses each the of modules in turn.

The Core package is the most fundamental part of the framework and provides the Dependency Injection features allowing you to manage bean container functionality. The basic concept here is the BeanFactory, which provides a factory pattern removing the need for programmatic singletons and allowing you to decouple the configuration and specification of dependencies from your actual program logic.

On top of the Core package sits the Context package, providing a way to access beans in a framework-style manner, somewhat resembling a JNDI-registry. The context package inherits its features from the beans package and adds support for text messaging using e.g. resource bundles, event-propagation, resource-loading and transparent creation of contexts by, for example, a servlet container.

The DAO package provides a JDBC-abstraction layer that removes the need to do tedious JDBC coding and parsing of database-vendor specific error codes. Also, the JDBC package provides a way to do programmatic as well as declarative transaction management, not only for classes implementing special interfaces, but for all your POJOs (plain old java objects).

The ORM package provides integration layers for popular object-relational mapping APIs, including JDO, Hibernate and iBatis. Using the ORM package you can use all those O/R-mappers in combination with all the other features Spring offers, like simple declarative transaction management mentioned before.

Spring's AOP package provides an AOP Alliance compliant aspect-oriented programming implementation allowing you to define, for example, method-interceptors and pointcuts to cleanly decouple code implementing functionality that should logically speaking be separated. Using source-level metadata functionality you can incorporate all kinds of behavioral information into your code, a little like .NET attributes.

Spring's Web package provides basic web-oriented integration features, such as multipart functionality, initialization of contexts using servlet listeners and a web-oriented application context. When using Spring together with WebWork or Struts, this is the package to integrate with.

Spring's Web MVC package provides a Model-View-Controller implementation for web-applications. Spring's MVC implementation is not just any implementation, it provides a clean separation between domain model code and web forms and allows you to use all the other features of the Spring Framework like validation.

Spring's web MVC framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale and theme resolution as well as support for upload files. The default handler is a very simple Controller interface, just offering a ModelAndView handleRequest(request,response) method. This can already be used for application controllers, but you will prefer the included implementation hierarchy, consisting of, for example AbstractController, AbstractCommandController and SimpleFormController. Application controllers will typically be subclasses of those. Note that you can choose an appropriate base class: If you don't have a form, you don't need a FormController. This is a major difference to Struts.

You can use any object as a command or form object - there's no need to implement an interface or derive from a base class. Spring's data binding is highly flexible, for example, it treats type mismatches as validation errors that can be evaluated by the application, not as system errors. So you don't need to duplicate your business objects' properties as Strings in your form objects, just to be able to handle invalid submissions, or to convert the Strings properly. Instead, it is often preferable to bind directly to your business objects. This is another major difference to Struts which is built around required base classes like Action and ActionForm - for every type of action.

Compared to WebWork, Spring has more differentiated object roles. It supports the notion of a Controller, an optional command or form object, and a model that gets passed to the view. The model will normally include the command or form object but also arbitrary reference data. Instead, a WebWork Action combines all those roles into one single object. WebWork does allow you to use existing business objects as part of your form, but only by making them bean properties of the respective Action class. Finally, the same Action instance that handles the request is used for evaluation and form population in the view. Thus, reference data needs to be modeled as bean properties of the Action too. These are arguably too many roles for one object.

Spring's view resolution is extremely flexible. A Controller implementation can even write a view directly to the response, returning null as ModelAndView. In the normal case, a ModelAndView instance consists of a view name and a model Map, containing bean names and corresponding objects (like a command or form, containing reference data). View name resolution is highly configurable, either via bean names, via a properties file, or via your own ViewResolver implementation. The abstract model Map allows for complete abstraction of the view technology, without any hassle. Any renderer can be integrated directly, whether JSP, Velocity, or any other rendering technology. The model Map is simply transformed into an appropriate format, such as JSP request attributes or a Velocity template model..

Pluggability of other MVC implementations

There are several reasons why some projects will prefer to use other MVC implementations. Many teams expect to leverage their existing investment in skills and tools. In addition, there is a large body of knowledge and experience avalailable for the Struts framework. Thus, if you can live with Struts' architectural flaws, it can still be a viable choice for the web layer. The same applies to WebWork and other web MVC frameworks.

If you don't want to use Spring's web MVC, but intend to leverage other solutions that Spring offers, you can integrate the web MVC framework of your choice with Spring easily. Simply start up a Spring root application context via its ContextLoaderListener, and access it via its ServletContext attribute (or Spring's respective helper method) from within a Struts or WebWork action. Note that there aren't any "plugins" involved, so no dedicated integration is necessary. From the web layer's point of view, you'll simply use Spring as a library, with the root application context instance as the entry point.

All your registered beans and all of Spring's services can be at your fingertips even without Spring's web MVC. Spring doesn't compete with Struts or WebWork in this scenario, it just addresses the many areas that the pure web MVC frameworks don't, from bean configuration to data access and transaction handling. So you are able to enrich your application with a Spring middle tier and/or data access tier, even if you just want to use, for example, the transaction abstraction with JDBC or Hibernate.

Features of Spring MVC

Spring's web module provides a wealth of unique web support features, including:

Clear separation of roles - controller, validator, command object, form object, model object, DispatcherServlet, handler mapping, view resolver, etc. Each role can be fulfilled by a specialized object.

Powerful and straightforward configuration of both framework and application classes as JavaBeans, including easy referencing across contexts, such as from web controllers to business objects and validators.

Adaptability, non-intrusiveness. Use whatever controller subclass you need (plain, command, form, wizard, multi-action, or a custom one) for a given scenario instead of deriving from a single controller for everything.

Reusable business code - no need for duplication. You can use existing business objects as command or form objects instead of mirroring them in order to extend a particular framework base class.

Customizable binding and validation - type mismatches as application-level validation errors that keep the offending value, localized date and number binding, etc instead of String-only form objects with manual parsing and conversion to business objects.

Customizable handler mapping and view resolution - handler mapping and view resolution strategies range from simple URL-based configuration, to sophisticated, purpose-built resolution strategies. This is more flexible than some web MVC frameworks which mandate a particular technique.

Flexible model transfer - model transfer via a name/value Map supports easy integration with any view technology.

Customizable locale and theme resolution, support for JSPs with or without Spring tag library, support for JSTL, support for Velocity without the need for extra bridges, etc.

A simple but powerful tag library that avoids HTML generation at any cost, allowing for maximum flexibility in terms of markup code.

Data Access using O/R Mappers

Spring provides integration with Hibernate, JDO, Oracle TopLink, Apache OJB and iBATIS SQL Maps: in terms of resource management, DAO implementation support, and transaction strategies. For example for Hibernate, there is first-class support with lots of IoC convenience features, addressing many typical Hibernate integration issues. All of these support packages for O/R mappers comply with Spring's generic transaction and DAO exception hierarchies. There are usually two integration styles: either using Spring's DAO 'templates' or coding DAOs against plain Hibernate/JDO/TopLink/etc APIs. In both cases, DAOs can be configured through Dependency Injection and participate in Spring's resource and transaction management.

Spring's adds significant support when using the O/R mapping layer of your choice to create data access applications. First of all, you should know that once you started using Spring's support for O/R mapping, you don't have to go all the way. No matter to what extent, you're invited to review and leverage the Spring approach, before deciding to take the effort and risk of building a similar infrastructure in-house. Much of the O/R mapping support, no matter what technology you're using may be used in a library style, as everything is designed as a set of reusable JavaBeans. Usage inside an ApplicationContext does provide additional benefits in terms of ease of configuration and deployment; as such, most examples in this section show configuration inside an ApplicationContext.

Some of the the benefits of using Spring to create your O/R mapping DAOs include:

Ease of testing. Spring's inversion of control approach makes it easy to swap the implementations and config locations of Hibernate SessionFactory instances, JDBC DataSources, transaction managers, and mapper object implementations (if needed). This makes it much easier to isolate and test each piece of persistence-related code in isolation.

Common data access exceptions.Spring can wrap exceptions from you O/R mapping tool of choice, converting them from proprietary (potentially checked) exceptions to a common runtime DataAccessException hierarchy. This allows you to handle most persistence exceptions, which are non-recoverable, only in the appropriate layers, without annoying boilerplate catches/throws, and exception declarations. You can still trap and handle exceptions anywhere you need to. Remember that JDBC exceptions (including DB specific dialects) are also converted to the same hierarchy, meaning that you can perform some operations with JDBC within a consistent programming model.

General resource management. Spring application contexts can handle the location and configuration of Hibernate SessionFactory instances, JDBC DataSources, iBATIS SQL Maps configuration objects, and other related resources. This makes these values easy to manage and change. Spring offers efficient, easy and safe handling of persistence resources. For example: Related code using Hibernate generally needs to use the same Hibernate Session for efficiency and proper transaction handling. Spring makes it easy to transparently create and bind a Session to the current thread, either by using an explicit 'template' wrapper class at the Java code level or by exposing a current Session through the Hibernate SessionFactory (for DAOs based on plain Hibernate3 API). Thus Spring solves many of the issues that repeatedly arise from typical Hibernate usage, for any transaction environment (local or JTA).

Integrated transaction management. Spring allows you to wrap your O/R mapping code with either a declarative, AOP style method interceptor, or an explicit 'template' wrapper class at the Java code level. In either case, transaction semantics are handled for you, and proper transaction handling (rollback, etc) in case of exceptions is taken care of. As discussed below, you also get the benefit of being able to use and swap various transaction managers, without your Hibernate/JDO related code being affected: for example, between local transactions and JTA, with the same full services (such as declarative transactions) available in both scenarios. As an additional benefit, JDBC-related code can fully integrate transactionally with the code you use to do O/R mapping. This is useful for data access that's not suitable for O/R mapping, such as batch processing or streaming of BLOBs, which still needs to share common transactions with O/R mapping operations.

To avoid vendor lock-in, and allow mix-and-match implementation strategies. While Hibernate is powerful, flexible, open source and free, it still uses a proprietary API. Furthermore one could argue that iBATIS is a bit lightweight, although it's excellent for use in application that don't require complex O/R mapping strategies. Given the choice, it's usually desirable to implement major application functionality using standard or abstracted APIs, in case you need to switch to another implementation for reasons of functionality, performance, or any other concerns. For example, Spring's abstraction of Hibernate transactions and exceptions, along with its IoC approach which allows you to easily swap in mapper/DAO objects implementing data access functionality, makes it easy to isolate all Hibernate-specific code in one area of your application, without sacrificing any of the power of Hibernate. Higher level service code dealing with the DAOs has no need to know anything about their implementation. This approach has the additional benefit of making it easy to intentionally implement data access with a mix-and-match approach (i.e. some data access performed using Hibernate, and some using JDBC, others using iBATIS) in a non-intrusive fashion, potentially providing great benefits in terms of continuing to use legacy code or leveraging the strength of each technology.

The Spring transaction abstraction

Spring provides a consistent abstraction for transaction management. This abstraction is one of the most important of Spring's abstractions, and delivers the following benefits:

Provides a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, iBATIS Database Layer and JDO.

Provides a simpler, easier to use, API for programmatic transaction management than most of these transaction APIs

Integrates with the Spring data access abstraction

Supports Spring declarative transaction management

Traditionally, J2EE developers have had two choices for transaction management: to use global or local transactions. Global transactions are managed by the application server, using JTA. Local transactions are resource-specific: for example, a transaction associated with a JDBC connection. This choice had profound implications. Global transactions provide the ability to work with multiple transactional resources. (It's worth noting that most applications use a single transaction resource) With local transactions, the application server is not involved in transaction management, and cannot help ensure correctness across multiple resources.

Global transactions have a significant downside. Code needs to use JTA: a cumbersome API to use (partly due to its exception model). Furthermore, a JTA UserTransaction normally needs to be obtained from JNDI: meaning that we need to use both JNDI and JTA to use JTA. Obviously all use of global transactions limits the reusability of application code, as JTA is normally only available in an application server environment.

The preferred way to use global transactions was via EJB CMT (Container Managed Transaction): a form of declarative transaction management (as distinguished from programmatic transaction management). EJB CMT removes the need for transaction-related JNDI lookups--although of course the use of EJB itself necessitates the use of JNDI. It removes most--not all--need to write Java code to control transactions. The significant downside is that CMT is (obviously) tied to JTA and an application server environment; and that it's only available if we choose to implement business logic in EJBs, or at least behind a transactional EJB facade. The negatives around EJB in general are so great that this is not an attractive proposition, when there are alternatives for declarative transaction management.

Local transactions may be easier to use, but also have significant disadvantages: They cannot work across multiple transactional resources, and tend to invade the programming model. For example, code that manages transactions using a JDBC connection cannot run within a global JTA transaction.

Spring resolves these problems. It enables application developers to use a consistent programming model in any environment. You write your code once, and it can benefit from different transaction management strategies in different environments. Spring provides both declarative and programmatic transaction management. Declarative transaction management is preferred by most users, and recommended in most cases.

With programmatic transaction management developers work with the Spring transaction abstraction, which can run over any underlying transaction infrastructure. With the preferred declarative model developers typically write little or no code related to transaction management, and hence don't depend on Spring's or any other transaction API.

References

[1] Rod Johnson.Expert one to one J2EE design and development.2003

[2] Martin Fowler.Inversion of Control Containers and the Dependency Injection pattern.20004

[3] Rob Harraop,Jan Machacek Publishing House of Electronles Industry.pro Spring.2004

[4] Craig Walls,Ryan Breidenbach.Manning Publications.Spring in action.2006

# 中文翻译

摘要：企业级解决方案中，天平正在由“重量级”体系如EJB，向轻量级框架如Spring倾斜。Spring 提供了大量实用化服务，它把许多极具竞争力的开源项目集成在一个框架中。在许多案例中，Spring框架正在取代传统J2EE体系，越来越多的开发者希望获得一些关于Spring的经验技巧。

关键词：轻量级框架 Spring MVC ORM

1.介绍：Spring 应用构架

当你第一次看到Spring的时候，你一定会说:"哦不，又一种web 构架".这篇文章将告诉你Spring明显区别于其他轻量级application framework, 它将专注于web的支持，与struts 和 webwork有着明显的区别。

在和struts 和 webwork的对比上,Spring是一个服务于所有层面的application framework:提供了bean的配置基础，AOP的支持，JDBC的提取框架，抽象事务支持，等等。它有一个非常显著的特点：在某个层面上如果你不需要Spring的支持，你就可以不使用String的class，只使用它的某一部分的功能。 从它的设计理念，你可以看到String 帮助你实现了真正的逻辑层和web层的分离：例如。一个校验应用将不用依靠controllers，就可以实现。这样的目标是 更好的重用和易测：过分依靠不必要的容器和框架将不能实现这一点。

当然,Spring 的自己的web支持和通常框架模式的细致完整.然而，Spring替换struts,webwork或者其他的web方案非常的容易.对于Spring的web支持或者不同的地方，Spring 允许你在web容器里面建立一个中间层，在测试环境或者标准独立的应用里面来设置重用你的商务逻辑.还有在J2EE环境里面，你的商务逻辑不必依靠容器提供的服务，像JTA ,EJB的支持.良好的构架的web应用可以运行在任何容器上，如，Tomcat 或者 Resin. 值得注意的是，Spring 不是和已经存在的解决方案进行竞争. 我们鼓励结合标准的技术，如， Servlet, JSP, JTA, JNDI, JDBC, and JDO, 和非常匹配的工具，如，Hibernate, Velocity, Log4J, and Caucho's Hessian/Burlap.这个框架的的设计思想是在你的应用需要改良的时候，你将会做一些技术的选择:例如,如果你需要分布式事务处理，你可能需要用Spring的Jta TransactionManager 来实现JTA服务.或者，用DataSourceTransactionManager or HibernateTransactionManager 来实现美妙完美的单个数据库交换。

2. Web MVC:Spring web 框架的设计思想

Spring 框架通过配置操作mappings,展示resolution,本地化和模版集成围绕着分派请求操作的servlet - DispatcherServlet设计的.缺省的操作是一个非常简单的控制接口, 他只提供了ModelAndView handleRequest(request,response)方法.这将用于一个应用的控制,但是,如果你想包含多个层次的控制，AbstractController, AbstractCommandController, MultiActionController, SimpleFormController, AbstractWizardFormController 将帮助你完成.应用控制将代表那些子系统.注意，你可以选择一个适当的基类:如果你没有 web form,你就不必用FormController.这就是和Struts最大的不同. 你可以通过命令或者form对象来操作任何对象:这不需要接口工具或者一个基础类的驱动.Spring的数据邦定是非常的灵活的.举例来说,它描述了具有在应用范围内的校验错误的输入机制,但不是系统错误.所以在你的form对象里面你不必复制你的业务对象的string属性,只操作出错的子任务,或者适当地转换string.换句话说,它通常可以很好的直接邦定你的业务对象. 这也是和struts围绕请求基础类Action 和 ActionForm (每个action操作类型) 建立主要的不同之一. 对比WebWork,Sping更多的区别在于对象角色:Sping支持控制器的感念,一个操作命令或者form对象,和得到数据传递给视图的模式.这个模式通常包含命令和form对象,但有时也包含任意的参考数据.换句话说,一个WebWork Action联合所有这些角色到一个单独的对象.WebWork 允许你用已经存在的业务对象作为你 form 的一部分,但是只生成各自Action 的 bean 属性. 最后,操作请求的Action 实例在一个视图里面获得付值和form population. 然而,参考数据也需要作为Action 的属性被模拟. 一个类里面有太多的角色是值得讨论的. 关于视图:Spring的视图方案非常的灵活. 一个控制器执行可以通过response 返回ModelAndView对象null,就可以直接写到一个视图.在通常的状况下,一个ModelAndView实例结合了 一个view 和一个 model Map,包含了bean name 和 通讯对象(像命令或者form,参考数据等等). View名称是非常高端的的配置，不是通过bean name, 一个properties 文件就是通过你自己的ViewResolver.这个抽象的model Map 允许你在视图层面完成提取,没有任何的争辩JSP/Velocity或者其他,每一种都可以直接完整使用.这个model Map 还可以简单得得到适当的格式化数据的转换,像JSP 请求属性或者Velocity 模版模式.

3.集成:用Spring一个不同web 框架

许多开发团队将为他们已有的项目或者新的项目已经获得的期限和工具进行投资.这里没有像Struts那样大量的图书和工具，但是同样我们有大量的拥有Spring开发技巧的开发人员.然而,如果你愿意生活在Struts的构架瑕疵中的话，他将是你在web层开发不错的选择.当然，其他应用也是一样. 如果你不想用Spring的 web MVC ，但是想借用Spring嫁接其他的解决方案,你可以非常简单地通过Spring 继承你自己的web 框架.你可以非常简单地通过ContextLoaderListener 启动一个Spring root application context, 并且,通过Struts 或者WebWork 的action 利用ServletContext 属性(或者Spring 的helper方法)存取它. 值得注意的是, 这里没有任何的 "plugins"被调用,因此没有专门的集成:来自web层的视图, 你可以简单的将Spring作为一个管理application context 实例入口点的类库. 所有你注册bean和Spring服务都可以在不需要Spring的web MVC的情况下都可以被你轻松掌握.Spring 不是和Struts ,WebWork这些应用进行竞争,它将作为一个纯web框架应用于 很多领域,因为它不需要配置数据存储和事务操作.所以,你可以利用Spring提供的中间层和数据存储层来丰富你的应用,甚至,你想用JDBC或者Hibernate进行事务抽象.

4.特点核对列表

如果聚焦于web 支持,Spring的一些显著特点是: 清楚地角色分离:controller , validator , command object , form object , model object,和 DispatcherServlet , handler mapping vs view resolver, 等等 强大而且直接的框架和作为JavaBeans的应用配置,包括简单的参照和应用内容,例如,从web控制器到业务对象和数据校验. 适应性,外挂:无论什么样的控制器你都需要得到代替Action/ActionForm所做的每件事情的方案(简单,命令,form,范例,多重action,或者定制一个) 重用业务逻辑代码,不需要复制:你可以用已经有的业务逻辑对象作为命令或则form对象代替反射特定的ActionForm子类. 可订制的邦定和数据校验:作为应用级的输入机制的错误校验,固定日期和数字邦定,例如,通过手动检查和转换业务对象来替换单个的string。 可订制的操作,可订制的视图方案:通过name/value Map灵活的模型传送,用一个简单的方法从简单到复杂操作映射和视图方案策略.

本地定制和主题方案:Spring没有taglib,完全支持于JSPs，JSTL,不需要额外的连接就可以支持Velocity,等等. 简单但是强大的tag library避免了HTML产生的混乱,最大限度的灵活扩展了标记代码.

5.关于spring 框架在web中的应用

Core包是框架的最基础部分并提供依赖注入(Dependency Injection)特性来使你可管理Bean容器功能。 这里的基础概念是BeanFactory，它提供Factory模式来消除对程序性单例的需要， 并允许你从程序逻辑中分离出依赖关系的配置和描述。

构建于Beans包上Context包，提供了一种框架式的Bean访问方式， 有些像JNDI注册。Context包的特性得自Beans包，并添加了文本消息的发送，通过比如资源串， 事件传播，资源装载的方式和Context的透明创建，如通过Servlet容器。

DAO包提供了JDBC的抽象层，它可消除冗长的JDBC编码和解析数据库厂商特有的错误代码。 该包也提供了一种方法实现编程性和声明性事务管理，不仅仅是针对实现特定接口的类， 而且对所有的POJO。

ORM包为流行的关系－对象映射APIs提供了集成层，包括JDO，Hibernate和iBatis。 通过ORM包，你可与所有Spring提供的其他特性相结合来使用这些对象/关系映射， 如前边提到的简单声明性事务管理。

Spring的AOP包提供与AOP联盟兼容的面向方面编程实现，允许你定义， 如方法拦截器和切点，来干净地给从逻辑上说应该被分离的功能实现代码解耦。 使用源码级的元数据功能，你可将各种行为信息合并到你的代码中，有点像.Net的attribute。

Spring的Web包提供了基本的面向Web的综合特性，如Multipart功能， 使用Servlet监听器的Context的初始化和面向Web的Applicatin Context。 当与WebWork或Struts一起使用Spring时，这个包使Spring可与其他框架结合。

Spring的Web MVC包提供了面向Web应用的Model-View-Controller实现。 Spring的MVC实现不仅仅是一种实现，它提供了一种domain model代码和web form的清晰分离， 这使你可使用Spring框架的所有其他特性，如校验.

Spring的web框架是围绕分发器(DispatcherServlet)设计的，DispatcherServlet将请求分发到不同的处理器，框架还包括可配置的处理器映射，视图解析，本地化，主题解析，还支持文件上传。缺省的处理器是一个简单的控制器(Controller)接口，这个接口仅仅定义了ModelAndView handleRequest(request,response)方法。你可以实现这个接口生成应用的控制器，但是使用Spring提供的一系列控制器实现会更好一些，比如AbstractController，AbstractCommandController，和SimpleFormController。应用控制器一般都从它们继承。注意你需要选择正确的基类：如果你没有表单，你就不需要一个FormController。这是和Structs的一个主要区别。

你可以使用任何对象作为命令对象或表单对象：不必实现某个接口或从某个基类继承。Spring的数据绑定相当灵活，例如，它认为类型不匹配这样的错误应该是应用级的验证错误，而不是系统错误。所以你不需要为了处理无效的表单提交，或者正确地转换字符串，在你的表单对象中用字符串类型重复定义你的业务对象属性。你应该直接绑定表单到业务对象上。这是和Struts的另一个重要不同，Struts是围绕象Action和ActionForm这样的基类构建的，每一种行为都是它们的子类。

和WebWork相比，Spring将对象细分成不同的角色：它支持的概念有控制器(Controller)，可选的命令对象(Command Object)或表单对象(Form Object)，以及传递到视图的模型(Model)。模型不仅包含命令对象或表单对象，而且也包含任何引用数据。但是，WebWork的Action将所有的这些角色都合并在一个单独的对象里。WebWork允许你在表单中使用现有的业务对象，但是只能把它们定义成不同Action类的bean属性。更重要的是，在运算和表单赋值时，使用的是同一个处理请求的Action实例。因此，引用数据也需要被定义成Action的bean属性。这样在一个对象就承担了太多的角色。

对于视图：Spring的视图解析相当灵活。一个控制器实现甚至可以直接输出一个视图作为响应，这需要使用null返回ModelAndView。在一般的情况下，一个ModelAndView实例包含视图名字和模型映射表，模型映射表提供了bean的名字及其对象(比如命令对象或表单对象，引用数据等等)的对应关系。视图名解析的配置是非常灵活的，可以通过bean的名字，属性文件或者你自己的ViewResolver来实现。抽象的模型映射表完全抽象了表现层，没有任何限制：JSP，Velocity，或者其它的技术——任何表现层都可以直接和Spring集成。模型映射表仅仅将数据转换成合适的格式，比如JSP请求属性或者Velocity模版模型。

MVC实现的可扩展性

许多团队努力争取在技术和工具方面能使他们的投入更有价值，无论是现有的项目还是新的项目都是这样。具体地说，Struts 不仅有大量的书籍和工具，而且有许多开发者熟悉它。因此，如果你能忍受Struts的架构性缺陷，它仍然是web层一个很好的选择。WebWork和其它web框架也是这样。

如果你不想使用Spring的web MVC框架，而仅仅想使用Spring提供的其它功能，你可以很容易地将你选择的web框架和Spring结合起来。只要通过Spring的ContextLoadListener启动一个Spring的根应用上下文，并且通过它的ServletContext属性(或者Spring的各种帮助方法)在Struts或WebWork的Action中访问。注意到现在没有提到任何具体的“plugins”，因此这里也没有提及如何集成：从web层的角度看，你可以仅仅把Spring作为一个库使用，根应用上下文实例作为入口。

所有你注册的bean和Spring的服务可以在没有Spring的web MVC下被访问。Spring并没有在使用方法上和Struts或WebWork竞争，它只是提供单一web框架所没有的功能，从bean的配置到数据访问和事务处理。所以你可以使用Spring的中间层和(或者)数据访问层来增强你的应用，即使你只是使用像JDBC或Hibernate事务抽象这样的功能。

Spring MVC框架的特点

如果仅仅关注于web方面的支持，Spring有下面一些特点：

清晰的角色划分：控制器，验证器，命令对象，表单对象和模型对象；分发器，处理器映射和视图解析器；等等。

直接将框架类和应用类都作为JavaBean配置，包括通过应用上下文配置中间层引用，例如，从web控制器到业务对象和验证器的引用。

可适应性，但不具有强制性：根据不同的情况，使用任何你需要的控制器子类(普通控制器，命令，表单，向导，多个行为，或者自定义的)，而不是要求任何东西都要从Action/ActionForm继承。

可重用的业务代码，而不需要代码重复：你可以使用现有的业务对象作为命令对象或表单对象，而不需要在ActionForm的子类中重复它们的定义。

可定制的绑定和验证：将类型不匹配作为应用级的验证错误，这可以保存错误的值，以及本地化的日期和数字绑定等，而不是只能使用字符串表单对象，手动解析它并转换到业务对象。

可定制的处理器映射，可定制的视图解析：灵活的模型可以根据名字/值映射，处理器映射和视图解析使应用策略从简单过渡到复杂，而不是只有一种单一的方法。

可定制的本地化和主题解析，支持JSP，无论有没有使用Spring标签库，支持JSTL，支持不需要额外过渡的Velocity，等等。

简单而强大的标签库，它尽可能地避免在HTML生成时的开销，提供在标记方面的最大灵活性。

使用ORM工具进行数据访问

Spring在资源管理，DAO实现支持以及实物策略等方面提供了与Hibernate, JDO和iBATIS SQL映射的集成。 对Hibernate，Spring使用了很多IoC的方便的特性提供了一流的支持，帮助你处理很多典型的Hibernate整合的问题。所有的这些都遵守Spring通用的事务和DAO异常体系.。

当您选择使用O/R映射来创建数据访问应用程序的时候，Spring的增加部分就会向您提供重要的支持。首先你应该了解的是，一旦你使用了Spring对O/R映射的支持，你不需要亲自作所有的事情。在决定花费力气，冒着风险建造类似的内部底层结构之前，我们都建议您考虑和利用Spring的解决方案。不管你使用的是何种技术，大部分的O/R映射支持都可以以library样式被使用，因为所有的东西都是被设计成一组可重复利用的JavaBeans。在ApplicationContext和BeanFactory中使用更是提供了配置和部署简单的好处，因此，这一章里的大多数例子都是在ApplicationContext中配置。

使用Spring构建你的ORM应用的好处包括:

测试简单. Spring的IoC使得很容易替换掉不同的实现，Hibernate SessionFacotory的位置，datasource, 事务管理, 映射对象的实现。这样就很容易隔离测试持久化相关代码的各个部分。

异常包装。 Spring能够包装Hibernate异常，把它们从专有的，checked exception变为一组抽象的runtime exception。这样你就可以仅仅在恰当的层处理大部分的不可恢复的异常，使你避免了很多讨厌的catch/throw以及异常声明。你还是可以在你需要的地方捕捉和处理异常。回想一下JDBC异常(包括与DB相关的方言)被转变为同样的异常体系，这就意味着你可以在一致的编程模型中处理JDBC操作。

通用的资源管理。 Spring的application context能够处理诸如Hibernate 的SessionFactory， JDBC的datasource，iBatis的SQLMaps配置对象以及其他相关资源的定位和配置。这使得这些配置的值很容易被管理和修改。Spring提供了有效，简单和安全的Hibernate Session处理。一般的使用Hibernate的代码则需要使用同一个Hibernate Session对象以确保有效和恰当地事务处理。而Spring让我们可以很容易透明地创建和绑定一个session到当前线程；你可以使用以下两种办法之一：声明式的AOP方法拦截器，或通过使用一个外部的template包装类在Java代码层次实现。这样，Spring就解决了在很多Hibernate论坛上出现的使用问题。

综合的事务管理 。 Spring允许你包装你的ORM代码，通过使用声明式的AOP方法拦截器或者在代码级别使用外部的template包装类。不管使用哪一种，事务相关的语义都会为你处理，万一有异常发生也会帮你做适当的事务操作(比如rollback)。就像我们下面要讨论的一样，你能够使用和替换各种transaction managers，却不会使你的Hibernate相关的代码受到影响。更好的是，JDBC相关的代码可以完全和Hibernate代码integrate transactionaly。这对于处理那些没有用Hibernate或iBatis实现的功能非常有用。

避免绑定特殊的技术，允许mix-and-match的实现策略。 虽然Hibernate非常强大，灵活，开源而且免费，但它还是使用了自己的特定的API。此外有人也许会争辩：iBatis更轻便而且在不需要复杂的O/R映射策略的应用中使用也很优秀。能够选择的话，使用标准或抽象的API来实现主要的应用需求，通常是更好的。尤其，当你可能会因为功能，性能或其他方面的原因而需要切换到另一个实现的时候。举例来说，Spring对Hibernate事务和异常的抽象，以及能够让你轻松交换mapper和DAO对象(实现数据访问功能)的IoC机制，这两个特性可以让你在不牺牲Hibernate性能的情况下，在你的应用程序中隔离Hibernate的相关代码。处理DAO的高层次的service代码不需要知道DAO的具体实现。这个方法可以很容易使用mix-and-match方案互不干扰地实现数据访问层(比如在一些地方用Hibernate，一些地方使用JDBC，其他地方使用iBatis)，mix-and-match有利于处理遗留下来的代码以及利用各种技术(JDBC,Hibernate,iBatis)的长处.。

Spring提供了一致的事务管理抽象。这个抽象是Spring最重要的抽象之一， 它有如下的优点：

为不同的事务API提供一致的编程模型，如JTA、JDBC、Hibernate、iBATIS数据库层 和JDO

提供比大多数事务API更简单的，易于使用的编程式事务管理API

整合Spring数据访问抽象

支持Spring声明式事务管理

传统上，J2EE开发者有两个事务管理的选择: 全局事务或 局部事务。全局事务由应用服务器管理，使用JTA。局部事务是和资源相关的：例如，一个和JDBC连接关联的事务。这个选择有深刻的含义。 全局事务可以用于多个事务性的资源(需要指出的是多数应用使用单一事务性 的资源)。使用局部事务，应用服务器不需要参与事务管理，并且不能帮助确保 跨越多个资源的事务的正确性。

全局事务有一个显著的不利方面，代码需要使用JTA：一个笨重的API(部分是 因为它的异常模型)。此外，JTA的UserTransaction通常需 要从JNDI获得，这意味着我为了JTA需要同时使用JNDI和JTA。 显然全部使用全局事务限制了应用代码的重用性，因为JTA通常只在应用服务器的环境中才能使用。

使用全局事务的比较好的方法是通过EJB的CMT (容器管理的事务)： 声明式事务管理的一种形式(区别于编程式事务管理 )。EJB的CMT不需要任何和事务相关的JNDI查找，虽然使用EJB本身 肯定需要使用JNDI。它消除大多数——不是全部——书写Java代码控制事务的需求。 显著的缺点是CMT绑定在JTA和应用服务器环境上，并且只有我们选择 使用EJB实现业务逻辑，或者至少处于一个事务化EJB的外观(Facade)后 才能使用它。EJB有如此多的诟病，当存在其它声明式事务管理时，EJB不是一个吸引人的建议。

局部事务容易使用，但也有明显的缺点：它们不能用于多个事务性资 源，并且趋向侵入的编程模型。例如，使用JDBC连接事务管理的代码不能用于 全局的JTA事务中。

Spring解决了这些问题。它使应用开发者能够使用在任何环境 下使用一致的编程模型。你可以只写一次你的代码，这在不同环境 下的不同事务管理策略中很有益处。Spring同时提供声明式和编程式事务管理。

使用编程式事务管理，开发者直接使用Spring事务抽象，这个抽象可以使用在任何 底层事务基础之上。使用首选的声明式模型，开发者通常书写很少的事务相关代 码，因此不依赖Spring或任何其他事务API。

参考文献：

[1] Rod Johnson .Expert one to one J2EE design and development.2003

[2] Martin Fowler.Inversion of Control Containers and the Dependency Injection pattern.20004

[3] Rob Harraop,Jan Machacek Publishing House of Electronles Industry.pro Spring.2004

[4] Craig Walls,Ryan Breidenbach.Manning Publications.Spring in action.2006