论文题目 在线客服聊天后台系统的设计与实现

专业学位类别	工 程 硕 士
学 号	201092231321
作 者 姓 名	倪万迁
指 导 教 师	刘贵松 副教授

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

在线客服聊天后台系统的设计与实现

（题名和副题名）

倪万迁

（作者姓名）

指导教师 刘贵松 副教授

电子科技大学 成 都

王 磊 高 工

济南旭景科技有限公司 济 南

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工程硕士

工程领域名称 软 件 工 程

提交论文日期 2014.3.25 论文答辩日期 2014.5.24

学位授予单位和日期 电子科技大学 2014 年 6 月 25 日

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

DESIGN AND REALIZATION OF BACKSTAGE OF ONLINE CUSTOMER CHATTING SYSTEM

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Major: **Master of Engineering**

Author: **Ni Wanqin**

Advisor: **Liu Guisong**

School : **School of Information and Software Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：_____ 日期：　　年　　月　　日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：_____ 导师签名：_____

日期：　　年　　月　　日

摘要

随着互联网的不断迅猛发展，越来越多的企业把自身运作、销售、管理都和网络结合起来了。互联网传播范围广，交互性强，信息量大，都大大地提高了企业的运营效率。在线客服就是互联网发展中诞生的一种帮助企业更方便与用户沟通的互联网工具。利用在线客服，企业可以通过网上与用户沟通，用户问题，避免额外设置线下人员与机构，降低企业运营成本，同时大大提高与客户的沟通效率。

在线客服通常有两种，一种是人工在线客服，以人工方式与用户在线上交流；一种是电子客服，以机器人形式与用户线上交流。而本文描述的在线客服则结合了这两种方式。当人工客服人手缺乏时可以切换到电子客服中，通过识别用户问题，利用后台知识库来对客户的问题进行简单的计算机自动应答。

本文就企业中的客户服务的社会背景进行了讨论和分析，提出了新时代的在线客服的运作方式，并阐述了新型在线客服的意义以及在企业与客户中的价值。接着指出了本文在线客服的人工在线客服和电子在线客服相结合的特点，分析并实现具有可扩展性、灵活性、可插入性的在线客服后台管理子系统。

本文按照软件开发的流程，详细阐述了在线客服的后台管理子系统的分析、设计和实现过程。描述了后台管理子系统以 B/S 模式，应用 Model、View、Control 分离的 MVC 模型来搭建系统总体架构，应用面向对象设计过程来设计整体类体系，利用 JavaEE 技术进行面向接口编程的分层次开发，利用 AJAX 异步调用来提高用户体验，结合 Struts2、Spring 和 Hibernate 等开源框架，实现了聊天记录查询，知识库管理，用户管理和客服统计共四个模块，最后对后台管理子系统进行简单测试，并给出运行效果。经检验，系统功能复合设计要求。

关键词：在线客服后台管理，知识库，聊天记录查询，JavaEE

ABSTRACT

With the continuous rapid development of Internet, more and more companies combine their enterprise operation, sales, management with network. Internet has a wide range of dissemination, powerful interactive, mass information, which have greatly improved the operational efficiency of enterprises. Online customer service is developed as a tool to help to improve the convenience of the communication between enterprise and customers during the development of Internet. With online customer service, companies can communicate with users vie the Internet to answer questions, avoid cost of additional human resource and institutions, while dramatically reduce operating costs and improving the efficiency of communications with customers.

There are two types of online customer service, one is artificial customer service while the other is as the type of online customer service robot. This article discusses the online customer service is a combination of both above. When there is lack of staff, the system can switch to robot. After identifying the user question, the robot uses the background knowledge base on the customer's question to a simple computer automatic response.

In accordance with the software development process, the article elaborates the analysis, design and implementation process of the background of online customer service management subsystem. In description, the background of management subsystem uses Brower/Server Mode; applies MVC model which makes Model, View, Control separated, to build overall system architecture; applies object-oriented design to design a whole class system; uses Java EE technology for interface-oriented programming to the hierarchical development; uses AJAX to make asynchronous calls in addition to the user experience. Also the subsystem is combined with Struts2, Spring, Hibernate and other open source frameworks to achieve the four modules: query of chat record, knowledge base management, user management, and customer service statistics. Finally, the article talks about the simple test of the subsystem, and gives operating results. After testing, the system function composite design requirements

Keywords： Online customer service, knowledge base, query of chat record, JavaEE

目录

第一章 绪论

1.1 课题研究背景

1.1.1 社会发展现状

进入 21 世纪，随着全球科技不断进步，人类的沟通方式也变得越来越高效了。科技进步带来的各种工具不断促进人们之间的交流。互联网就是当前发展最为迅猛的一种知识分享、交流工具。尤其最近几年的发展，互联网更是从 1.0 时代一跃发展成为 2.0 时代，其传播范围更广，交互性更强，信息交互更自由，内容更丰富，更重要的是——人人都可以是信息的提供者，这都大大拓展了人们的知识获取的广度、深度和速度。

在科技迅猛发展的同时，企业之间的竞争也越来越激烈。企业之间的竞争不但表现在产品上，还表现在服务上。当前市场环境下，商品同质化越来越严重，企业都把目光放到了服务上面。在质量与价格都差不多的情况下，消费者定会选择服务质量更好的企业。所以，很多企业都纷纷把制高点放在了商品服务上。如何提高企业的服务质量，增强企业的竞争优势，都是企业关注的焦点。而其中，客户服务又是产品服务中一种重要形式，其重要性可想而知。

客户服务在商业实践中一般会分为三类，即：售前服务、售中服务、售后服务。售前服务一般是指企业在销售产品之前为顾客提供的一系列活动，如市场调查、产品设计、提供使用说明书、提供咨询服务等。售中服务则是指在产品交易过程中销售者向购买者提供的服务，如接待服务、商品包装服务等。售后服务是指凡与所销售产品有连带关系，并且有益于购买者特征的服务，主要包括送货、安装、产品退换、维修、保养、使用技术培训等方面的服务。客户服务通常是通过电话进行，但也可以通过电子邮件、聊天、传真、自服务或邮件进行[1,2]。

1.1.2 客户服务现状

当前，很多企业都还只是停留在通过人工客服与客户间的电话联系、或者通过咨询台与客户进行面对面交流。这种传统方式虽然可行，但是在不同的情况下却会凸显出不同的缺点。

(1) 当企业刚起步或者资源紧缺时，通过传统方式来提供客户服务带来的成本相对较高，企业难以维持。呼叫中心、咨询服务点之类的传统客户服务方式所需要的硬件资源、人力资源都是相当可观的，在企业资源不足的时候难以负担。

即使单单通过电话的方式来为客户提供服务也会带来一定成本，而且效率不高，智能化程度低。

(2) 当企业知名度越来越高，客户交流的需求越来越频繁的时候，企业需要增加客服人员来解决不断增长的客户咨询数所带来的压力。这样必然会增加企业的管理成本，人力资源成本以及相关设备带来的运营成本。从这个角度出发企业的竞争力必然会降低。

(3) 当客户咨询数上升而不增加客服人手，则会增大客服人员工作压力。反映在工作效果上必然是回复客户不及时或回复质量不高。这样必然给客户的影响不好，给企业降低了服务质量。

1.1.3 解决思路

互联网技术的发展大大方便了人们的交流，而企业客户服务的传统实现方式的弊端也是在沟通交流上，那是不是可以尝试利用互联网技术来改善传统的客户服务方式呢。互联网优势在于成本低、时效性强、覆盖面广，把互联网技术作为新鲜血液注入到传统客户服务上必将弥补后者的多种缺点，让企业的服务做得更周到，让客户体验服务更满意。

如何降低客户服务的成本，同时又提高对客户咨询的响应速度和质量，在传统的客户服务中是比较难实现的。这就需要一种在保证响应质量前提下低成本的快速相应客户的一种回复手段，互联网技术的发展使这样的需求成为了可能。在线客服就是在这种环境下提出的。

在线客服就是互联网发展中诞生的一种帮助企业更方便与用户沟通的一种互联网工具。利用在线客服，企业可以通过网上与用户沟通，解答用户问题，避免额外设置线下人员与机构，降低企业运营成本，同时大大提高与客户的沟通效率。

1.2 在线客服的意义与本系统特点

在线客服以网页为载体，运用最新网络技术为网站访客提供与网站客服即使通讯的高科技手段[2]。以网络方式来与客户进行沟通对客服工作而言是简单而高效的，因为客户只需要登陆企业网站访问客服页面，就可以与客服工作人员进行交流了，其中客户并不需要安装其他软件，只需要浏览器和网络即可。对企业而言成本也是相当低廉的，因为企业只需要在原有的网站系统上增加在线客服软件系统即可。这样就解决了企业客服中的设备成本问题。

更进一步，在线客服也可以解决人力资源问题，可以让企业减少客服工作人

员，这是通过电子在线客服来实现的。电子在线客服是一种自动客服，自动客服可以自动回答客户的问题，是一种非常重要的客服方式。在企业的相关信息发布阶段，管理者将企业中常见的问题按一定的组织方式，存放到知识和信息库中。当客户在遇到疑难问题是，通过自动客服提交的描述，系统将根据客户提交的问题进行相应地处理，对知识和信息库进行自动搜索或语义匹配，按照检索内容的相关程度，将该问题或相关问题的解答呈现给客户。当匹配的相关程度不够高时，可以认为没有找到符合的答案，将该问题保存到数据库中，留待只是管理者给出相应的答案。随着知识库的不断完善，自动客服便能够解决越来越多的问题[3]。

所以，通过电子在线客服就可以把问题映射到知识库中，自动地把知识库的问题的对应答案返回给客户，做到在线客服的无人智能回复，降低了企业的人力资源成本。

本文中开发的在线客服系统就是属于结合了人工在线客服系统和电子在线客服的一种新型在线客服系统。人工在线客服，即企业安排人力资源进行与客户一对一或一对多的进行点对点交流。这种方式优点是企业与用户直接交流，能与客户进行有效的沟通交流，但是缺点也很明显，就是当客户并发多的时候没有足够的人工客服进行服务。这就导致客户需要排队接受服务，甚至有些客户根本就不能得到服务。而电子客服可以同时发起很多个电子客服来对大量客户进行服务，对客户的请求能及时响应，减少客户的等待时间，提高用户体验。当时缺点也很大，就是很多时候不能很好地理解客户输入的内容，导致与客户无法很好的沟通，这是当前技术发展的一个瓶颈。所以不能依靠电子客服来完全取代人工客服。因此，针对如何在企业资源与服务质量找到一个平衡点，本系统提出了将人工在线客服和电子在线客服融合在一起的解决方案。

同时，本系统可提供一个后台管理系统来维护知识库。可以通过后台子系统来对知识库进行维护，并支持聊天记录查询，这样可以从聊天记录中提取有效信息来更新知识库，让电子在线客服服务的质量更好。

1.3 本文主要内容

本文在分析在线客服的需求的基础上，简要介绍在线客服的主系统的设计及功能点，再详细分析、设计并实现一个支持主系统的后台管理子系统。本文的主要内容有：

(1) 包括业务对本系统的需求和主系统对后台知识库管理子系统的需求。尤其对后台子系统的系统作进行详细的需求分析。明确本文中后台管理子系统的设

计目标。

(2) 对客服聊天系统作总体的体系架构分析，介绍相应技术背景知识，并重点研究后台知识库管理子系统的系统构架。分析并设计客服务问题知识库，并给出一个易于维护知识库的解决方案。分析并设计在线客服聊天记录的储存方式，以便方便对历史聊天记录的查询及再利用。分析并设计在线客服的人工在线客服部分的客服工作量统计。

(3) 对后台子系统进行详细设计的描述，包括数据库模型设计，给模块功能设计（接口设计，类层次设计，流程逻辑设计等），系统间交互设计等等。

(4) 对主要的类进行代码详述，并展示系统成果（界面展示）。对系统测试方案进行概述，包括系统单元测试，集成测试等。

本文的目的就是要分析并实现这样一个具有可扩展性、灵活性、可插入性的在线客服后台管理子系统。

1.4 论文的组织结构

全文的章节结构安排如下：

第一章为绪论，简要介绍在线客服聊天后台聊天系统来源背景、发展现状及研究内容。

第二章为后台聊天系统的相关技术介绍。

第三章为在线客服聊天后台聊天系统的需求分析。

第四章为在线客服聊天后台聊天系统的设计，系统分析，在线客服聊天后台系统的详细设计，在该系统需求分析和概要设计的基础上进行进一步的详细设计，划分功能模块并对功能模块进行功能设计。

第五章是在线客服聊天后台系统的功能实现，针对系统的需求进行系统功能模块的实现，在该系统需求分析和概要和详细设计的基础上进行进一步的功能实现和系统编程实现工作。

第六章是系统测试，针对系统需要用到功能进行的测试，给出了系统的测试方法及结论。

第七章是总结和展望，对自己论文课题工作进行全文总结，并对下一步的工作进行展望。

第二章 系统相关技术简介

2.1 后台管理子系统架构图

由于后台管理子系统基本属于信息管理系统，所以本文讨论的该子系统决定采用基于 Java 平台的 MVC 模式的 Web 开发方式，在表现层采用 Struts2 框架，业务层用 Spring2 容器来管理，数据持久层用 Hibernate3 框架来操作数据库。系统采用以 Eclipse3.5 JavaEE 版作为为主开发环境、 Mysql5 为数据库管理系统、tomcat5.5 为服务器，通过 Maven2 来作为管理项目。系统架构图如图 2-1 所示：

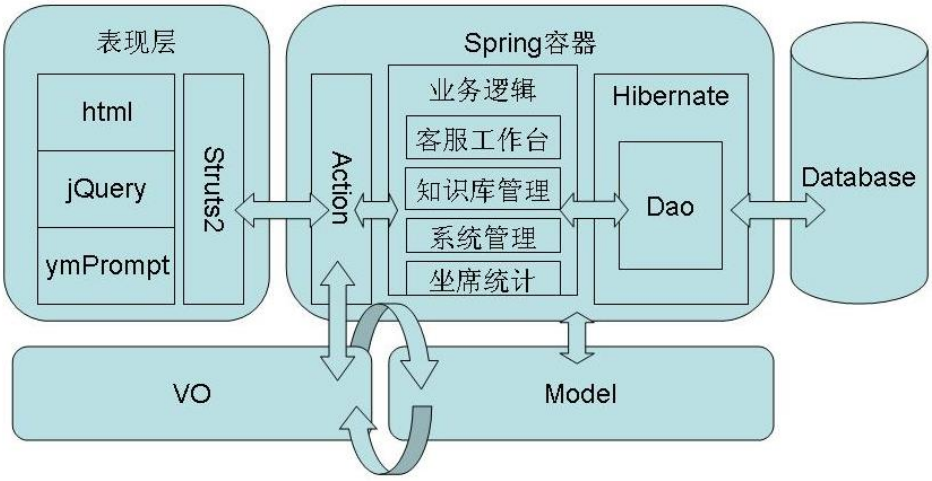

图 2-1 后台管理架构图

2.2 后台管理子系统相关技术简介

2.2.1 Java EE

Java EE 以前称为 J2EE，是一种应用程序，其作用主要在于帮助开发和部署，简单来说，Java EE Java 应用程序，其特点是是可移植的、健壮的、可伸缩且安全的服务器端应用程序。Java EE 是在 Java SE 的基础上构建的，它提供 Web 服务、组件模型、管理和通信 API，可以用来实现企业级的面向服务体系结构（SOA）和 Web 2.0 应用程序。

Java Web 技术因其后面有强大的 Java EE 技术做支持，在开发大型的、企业级的应用程序中将成为首选。在 Java Web 应用中包含如下内容：Servlet、Jsp、实用类、静态文档，如 HTML、图片等。构成 Java Web 应用的最主要的组件就是 Servlet 和 Jsp。但直接使用单纯采用 JSP 实现页面逻辑，系统业务逻辑与页面显

示混合，在将来业务逻辑发生变更时维护将变得困难；其次，采用自定义的 Servlet 组织流程，标准不一致也会为以后的维护留下隐患[4, 5, 6]。所以技术上我们选用了多层体系结构（Struts 框架，Spring 框架，Hibernate[7, 8, 9]），其中有 Struts 框架基于 MVC 设计模式， Spring 框架提供 IOC 功能。

2.2.2 MVC 模式

MVC 即模型——视图——控制器(Model-View-Control)。

MVC 架构有助于将应用程序分割成若干逻辑部件，使程序设计变得更加容易。

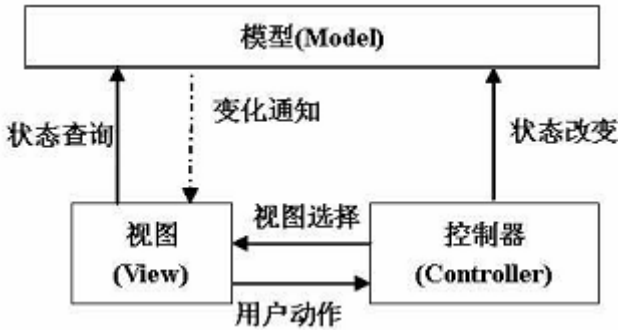

图 2-2MVC 模式示意图

标准的 MVC 设计模式中，用户一旦发出请求以后会将所有请求交给控制层处理，然后由控制层来调用模型层中的模型组件，并通过这些组件来进行持久层的访问，再将所有的结果都保存在 JavaBean（Java 类）中，最终由 JSP 和 JavaBean 来一起完成页面的显示。

2.2.3 AJAX 技术

AJAX 通常需要自己写底层的 Javascript 代码来实现，当系统应用 AJAX 很多时会导致相关代码零乱难以组织，不利于后期扩展，所以现在出现很多 Javascript 框架集成了 AJAX，例如本系统经常用到的 JQuery 框架。使用 jQuery 将使 Ajax 变得及其简单。jQuery 提供有一些函数[10, 11, 12]，可以使简单的工作变得更加简单，复杂的工作变得不再复杂。

采用面向服务的建模方法，构建所有业务逻辑的五种服务模型：工具服务、实体服务、任务服务、流程服务和界面服务，形成考务业务服务资产库，作为系统实现的服务资源库；

采用三层架构思想，将业务服务资产库部署为中间业务逻辑层，将系统的表现层与业务逻辑层分离，将业务逻辑层与数据层分离，构建一个高度灵活、可扩展、可伸缩的系统技术架构；

采用 Web 瘦客户端体系架构，对于 B/S 子系统的 Web 客户端，不涉及系统业务逻辑的实现，只起到系统表现层的作用，完成信息的输入/输出工作，满足客户端免维护、零配置的要求；

采用集中的消息队列实现数据交换，采用消息队列中间件满足系统之间数据交换的需要，避免采用点到点的两两数据交换接口的实现模式，既支持发布/订阅方式的异步消息交换模式，又支持请求/应答方式的同步消息交换模式。通过消息队列中间件满足数据交换的可靠、安全和高效。

2.2.4 队列

队列是一种特殊的线性表，它只允许在表的前端(front)进行删除操作，只允许在表的后端（rear）进行插入操作。队列是设计程序中常用的一种数据结构。数据元素只能从队尾进入，从队首取出。在队列中，数据元素的次序不会改变。每当有数据元素从队列中被取出，后面的数据元素依次向前移动一位。

java 中 LinkedList 实现了 Queue 接口，所以，系统实现时可以把 LinkedList 当成 Queue 来用。

2.2.5 UML 可视化建模概述

UML 是统一建模语言，是可视化通用的建模语言，它支持大部分现存的面向对象开发过程 [13, 14]。

UML 划分各种概念和组件的方法是视图。视图也就是 UML 建模组件的子集，该子集能够表达系统某一方面特征。由于本文主要涉及了类图和活动图，所以下面就简述这 2 种。

(1) 类图。

类图属于静态视图。静态视图描述的系统行为与时间无关，类图的类用矩形框表示，属性和操作在分格中（如果不需要详细信息，分格可以省略），关系用类框之间的连线来表示，不同的关系用连线上和连线端头处的修饰符来区别。

UML 中的大部分视图都可不断进行类似的细化[14]。

(2) 活动图。

活动图是动态视图的一种，描述执行算法的工作流程中涉及的活动，描述了一组顺序的或并发的活动。活动状态代表了一个活动的一个工作流步骤或一个操作的执行。活动视图用活动图来体现[14]。

活动图显示从活动到活动的流。一个活动是一个状态机中进行的非原子的执行单元。活动的执行最终延伸为一些独立动作的执行，每个动作将导致系统状态

的改变或者消息传送。动作包括调用另一个操作，发送一个信号，创建或销毁一个对象，或者某些纯计算（例如对一个表达式求值）。在图形上，活动图是点点和弧的集合[15]。

2.3 本章小结

本章阐述了后台管理子系统架构描述，为实现系统奠定了技术方向。再简单介绍了后台子系统涉及到的相关技术。下一章将在系统分析的基础上进行后台管理子系统的详细设计。

第三章 系统需求分析与获取

3.1 在线客服系统整体组成

如图 3-1 所示，系统主要由以下几部分组成：用户端、客户端、主系统（电子客服模块、人工客服模块、仲裁模块）、后台管理子系统、数据库组成。

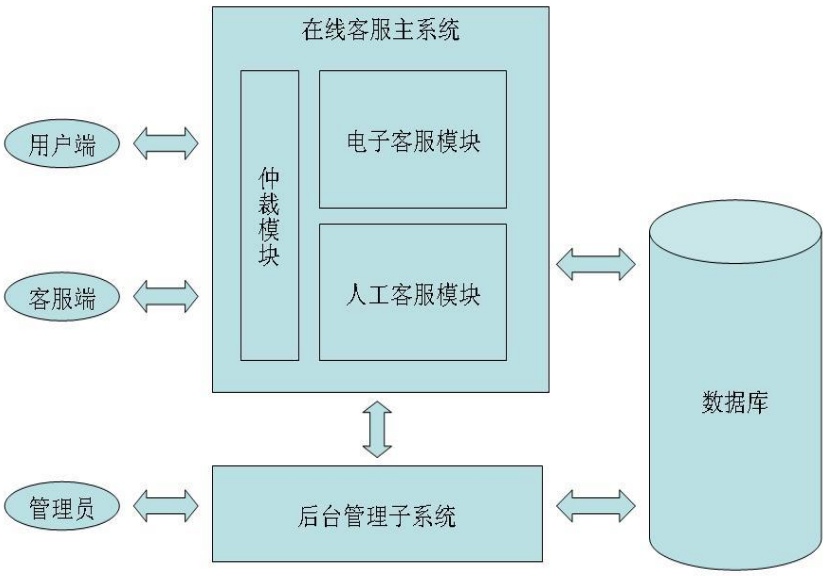

图 3-1 在线客服整体组成示意图

(1) 在线客服主系统主要负责接收用户的问题和返回答案。主系统主要包括仲裁、电子客服、人工客服三个模块。仲裁模块主要负责分配用户会话给电子客服或人工客服。它判断人工客服的负荷是否饱和，如果饱和则将用户会话分配给电子客服，否则就按一定规则分配给指定客服工作人员。电子客服模块实现了对客户问题的自动回答，它主要基于 Lucene 全文信息检索框架，先对客户问题进行分词处理，通过已有的知识库索引检索知识库问题，再把答案返回给用户。人工客服模块为客服端提供了支持，为客服工作人员与客户的正常沟通提供了保障，包括向客服人员转发用户问题和接收客服输入内容、提供常用语选择等服务，是与客户端通讯的桥梁。

(2) 用户端主要为用户提供了操作界面。用户可以通过用户端输入问题和接

受系统返回来的答案，在异步请求交互技术的支持下用户可以很方便地与在线客服系统进行局部刷新的聊天功能，具有良好的用户体验。在用户端中输入问题系统还会及时根据当前输入来返回最匹配的已有问题的列表，这样用户就可以列表选择问题或者继续补全问题，方便用户输入和有助于用户准确表达问题，提高效率。

(3) 客服端主要为客服人员提供了操作界面。客服工作人员进入客服端页面就可以看到系统已为他分配的用户会话，客服人员通过选择某个用户会话来同用户进行交流。客服人员还可以进行勾选待入库问题来标识有可能需要加入到知识库里面的问题，可以选择常用语来快捷回复客户问题，可以查看聊天记录，还可以选择把会话转移到电子客服中去。

(4) 后台管理子系统为主系统运行提供了支持。后台管理中包括了知识库维护、聊天记录查询、人员管理、客服统计等内容。通过后台管理，结合聊天记录查询，知识库的内容可以不断扩充，可以为电子客服解答用户问题提供更多的内容。人员管理则提供了在线客服系统相关人员的登陆、权限、信息修改的管理功能。客服统计可以更好的帮助分析人工客服的工作情况。

(5) 数据库中承载着整个系统运行的数据。包括知识库问答数据，聊天记录，系统日志，人员信息，同义词库等等。

3.2 在线客服主系统需求概述

3.2.1 系统应用背景

由于当今社会市场经济竞争日趋激烈，不少企业已经从当初单纯的产品的品质、价格竞争逐步转移到了服务的竞争。同时消费者的消费观念不断提升，对服务质量的要求越来越高，优质的售前、售中和售后服务是企业打响品牌的关键。所以，很多企业都纷纷加大了客户服务的工作力度，不断提高服务水平。

现在多数企业打多采用 Email、BBS、FAQ、人工客服、在线客服等方式来实现对客户的咨询服务[3]。如果只使用传统方式的客户服务，那对设备和人力资源的投资将会很大；如果使用网络方式的在线客服，由于资源有限，不可能根据客户数来安排人工客服，即使是人工客服一对多个客户，也不能解决大量客户并发的问题；如果只使用电子客服，效果也不理想，因为电子客服的能力有限，只能在一定范围里面协助人工客服来解决问题。因此，如何在企业资源与服务质量找到一个平衡点是一个很大的问题。

所以整合人工在线客服与电子在线客服就自然而然地成为理想的选择。本系

统可以在这两者间协调。在人工客服资源充足的情况下提供人工客服与上网用户交流的平台，当人工客服资源不足时，系统会自动提供电子客服平台与上网用户交流，以弥补人工客服资源不足情况。

3.2.2 系统业务分析

本系统的客户服务结合了人工服务和电子服务来回复客户的咨询，同时可以按一定要求在人工和电子服务之间切换，力求在保证服务质量的前提下使企业在客服工作的成本尽可能减少。

如图 3-2 所示，图片展示了客户如何与客服人员、电子客服进行交互。

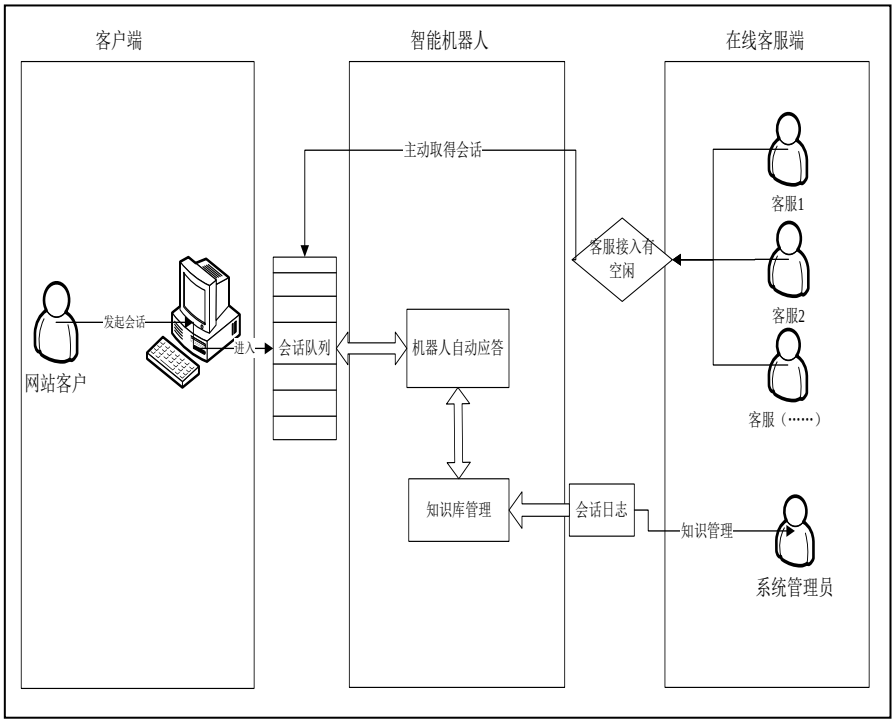

图 3-2 在线客服客户、人工客服、电子客服交互示意图

结合该图分析系统业务概要流程：

(1) 网站客户通过点击网页中在线客服连接进入咨询界面；

(2) 系统将该客户加入到会话队列中，供后续人工客服或电子客服获取；

(3) 客户通过网页输入咨询问题，系统可以根据客户输入动态列出与客户输入问题最接近的问题供客户选择，客户输入完毕或选择已列出的问题后提交到服务器中；

(4) 当客服工作人员人手充足时，主动从会话队列获取会话，并开始与客户对话；

(5) 客服工作人员可以在对话过程中勾选某条聊天记录，以此标记为待入库

问题，便于以后知识库的更新与维护；

(6) 当客服工作人员人手缺乏时，机器人自动应答，由机器人来回答：系统通过语义分析客户输入问题，把分析结果与知识库的索引文件相关联，搜索知识库中的问题答案，然后回答客户；

(7) 系统管理员可以通过知识库管理来维护知识库（系统管理员可以查看待入库问题以遴选问题来更新知识库）

3.2.3 系统总体用例图

系统总体用例图如图 3-3 示：

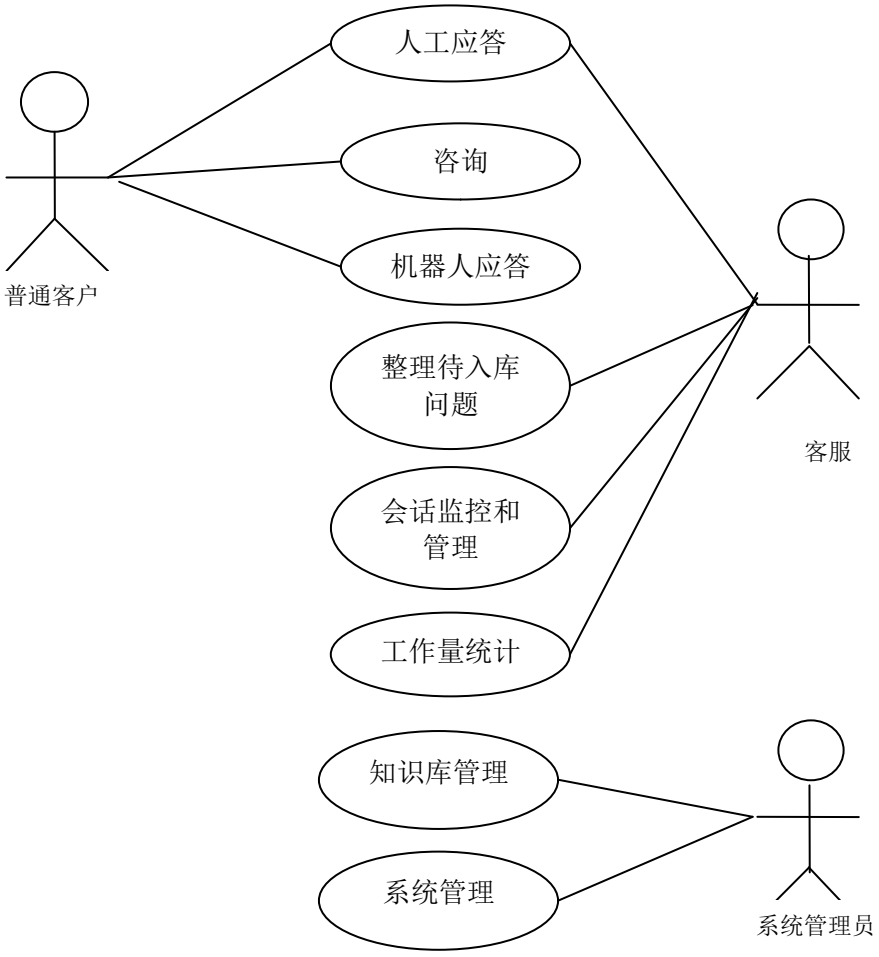

图 3-3 系统总体用例图

3.3 在线客服后台管理子系统需求分析

本文的在线客服系统涉及了电子客服，这必然需要有一个后台来维护知识库，管理在线客服聊天系统的信息。根据主系统运行需要，后台子系统主要功能包括对聊天系统进行聊天数据记录，聊天记录查询，聊天内容知识库管理这几方

面的功能，为主系统提供一个易于操作，科学管理的后台解决方案。

经过需求分析，可总结出后台管理子系统主要分为以下四个模块：

(1) 会话监控及会话记录查询。客服人员在系统中实现对各种状态的会话进行监控和管理。客服可以查询当前进行中的实时会话或者过去结束了的会话历史记录，所查询的会话记录可以是电子客服或者是人工客服的。可以查看会话的详细信息，客服人员可以对待处理的会话进行应答回复以启动人工客服。

用例图如下图 3-4 所示：

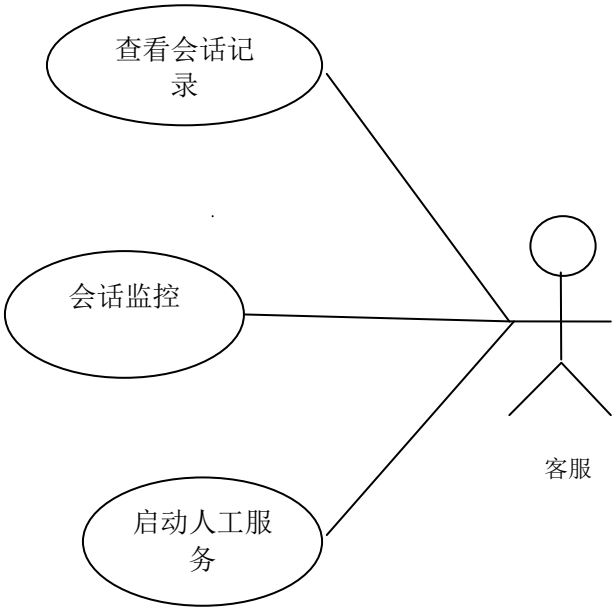

图 3-4 会话监控用例图

(2) 知识库管理。管理人员实现对知识库中的知识架构进行组织和管理。管理人员可以对业务类型、问题分类、具体问题进行新增、修改、删除。其中问题的增删改查涉及了问题标题、内容和关键词、相应 url 等内容。对于知识库的查看，需要通过树形目录来组织以方便查看，

知识库架构分三层，第一、二层是问题分类，第三层是具体问题，且问题只能出现在第三层中，知识库架构图如图 3-5 所示。

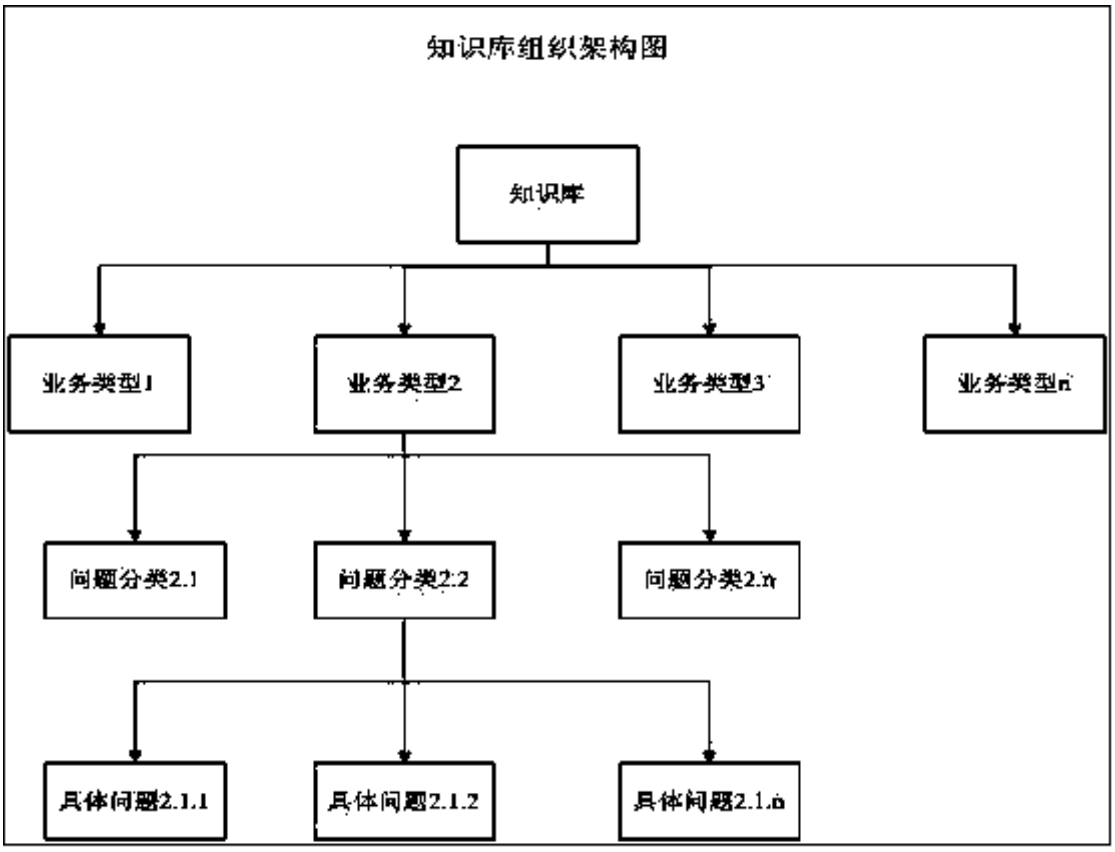

图 3-5 知识库组织架构图

用例图如下图 3-6 所示：

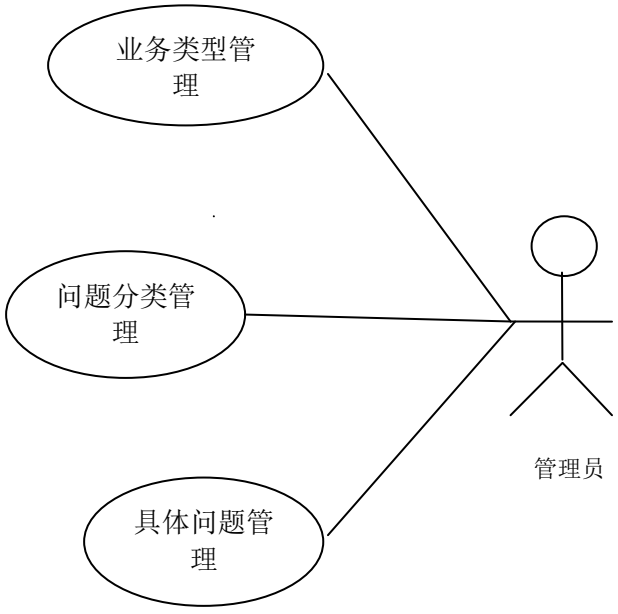

图 3-6 知识库管理用例图

(3) 用户管理。系统管理员能对系统的用户进行增删改查的操作，能对当前的系统管理员进行密码修改，能查询所有用户列表信息。使用本系统的用户角色有系统管理员，普通客服，高级客服，知识库管理员和监控客服。在用户管理模块中能对用户基本资料、权限角色等资料进行基本修改操作。

用户管理用例图如下图 3-7 所示：

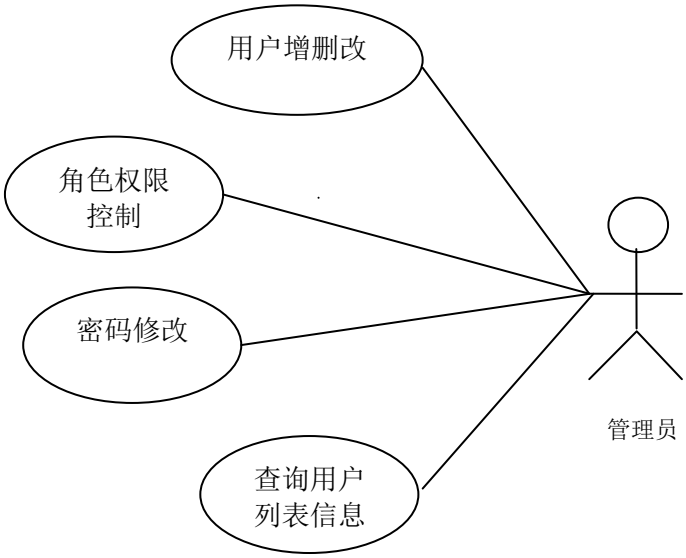

图 3-7 用户管理用例图

(4) 工作量统计。客服管理人员统计某日期范围内某一客服人员的工作量。并能导出到 excel 表格中。统计内容有已答会话总数，未答会话总数，总转移会话数。

工作量统计用例图如图 3-8 所示：

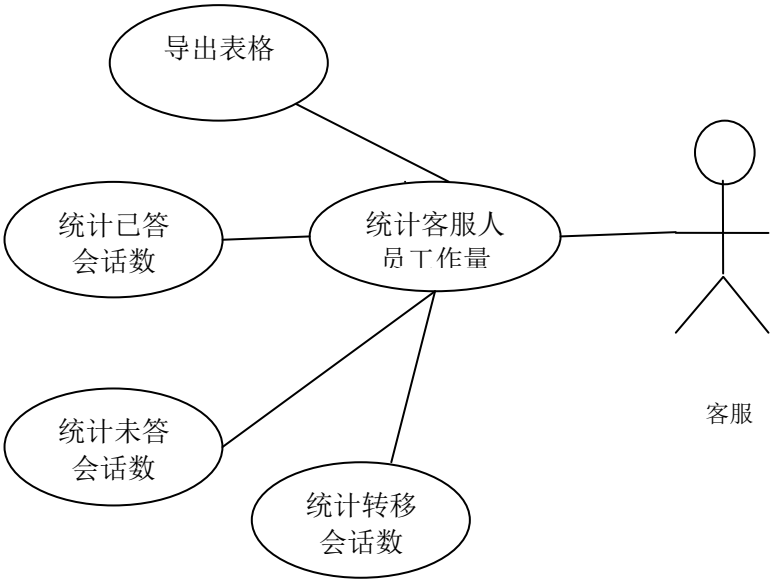

图 3-8 工作量统计用例图

(5)安全、可靠与保密需求

数据安全要求

要求系统采用服务器托管、双机热备份、磁盘阵列等多种措施，保障数据安全可靠。

要求建立数据容灾备份机制，在意外事故或灾难发生时进行数据保护。

网络安全要求

要求配置硬件防火墙系统，具有网络数据监控、网络嗅探等功能，可以方便实现对非法报文进行阻断。

要求配备防病毒工具，支持邮件、系统等的实时扫描和病毒防护，支持动态病毒特征库的更新，提供集中式网络防病毒管理。

要求配置入侵监测系统，对黑客等恶意攻击，实时监测，保护系统安全。

系统可靠要求

在大数据量并发访问时，要求系统支持数万级用户并发访问。

数据保密要求

系统建设及技术支持过程中涉及到大量的保密信息，包括聊天信息、客户信息等，要求接触这些信息的服务人员是可信的。需要制定周密的保密措施，确保信息的保密。

系统容错要求

要求系统具有良好的容错能力，程序出现异常情况时，不对系统和数据造成破坏，并以适当方式提示给用户解决办法。

3.4 非功能性需求

扩展性：保证系统在添加新功能点情况下仍能够正常运行，而不会导致其他模块出现问题，是系统具备稳定扩充能力。

灵活性：保证系统在对某些模块进行修改后仍能正常运行，而不会导致其他模块出现问题，保证系统对变化适应能力。

插入性：保证系统在对某些模块进行替换后仍能正常运行，而不会导致其他模块出现问题，保证系统对模块变更稳定性。

易用性：为用户提供良好的用户体验，用户能方便使用系统，降低用户对系统的学习成本。

可用性：保障系统能长时间无故障运行，需要在 24*7 小时的环境中运行。

响应时间：用户通过电子客服进行咨询时，在 95%的情况下，一般响应时间不操作 1 秒，在通过人工客服进行咨询时，在 90%的情况下，一般响应时间不超

过 20 秒。

系统性能指标要求如下：

1.终端并发指标

三千人以上同时后台聊天会话发起申请；

三十人以上人同时在线聊天，其他人在队列等待；

2.系统负载指标

CPU 小时内平均负载<35%；

应用服务内存小时内平均<55%。

3.界面操作响应时间

针对用户提供的查询服务响应时间<1.5s；

查询 1000 条内信息<10s；

增删改简单业务操作提交响应时间<2s；

对于其他耗时较长的操作，给出相应进度提示，不允许出现白屏现象。

4.可靠性指标

数据库双机热备避免单机故障；

应用服务器集群部署避免单机故障；

Web 服务器集群部署避免单机故障；

网络设备、网络接入全双备避免单点故障；

数据中心支持多重备份，确保数据可靠；

相关数据信息等存储多份、多点存储保障断电、断网、单机系统崩溃情况下不会丢失信息。

5.安全指标

聊天内容等个人隐私的获取需要经过授权，对外发布信息不允许涉及个人信息；

机构间数据传输采用 VPN、HTTPS 等安全通道；

操作人员可采用基于硬件 USBKEY 的身份认证机制，确保操作人员身份；

重要操作采用日志＋数字签名机制进行记录，确保有权限人员的操作可追踪。

6.适应性、扩展性指标

采用面向服务的架构（SOA）设计和实现，容易实现服务功能组合及与外部的接口，扩展性强；

主体业务引入业务流程管理，适应业务流程的变化；

针对业务功能环节进行设计，形成组件，业务独立性强，容易维护和扩展。

7.软件开发能力需求

1）系统开发计划

为保证系统开发有序进行，要求系统提供明确的项目开发计划书，包括组织结构及相关责任、实施人员资历和分工（详细到人）等。

2）系统开发和管理工具

要求方案中说明系统开发和管理相关工具，包括：开发工具、版本管理工具、对象分析与建模工具、数据库建模工具、文档管理工具、测试管理工具、项目管理工具等。

3）系统测试

要求系统开发过程中配备专门测试人员负责测试工作，并有相应的测试文档，包括测试计划、测试用例和说明、测试报告以及问题和更改记录等。

4）风险识别与规避

要求能够识别本项目建设和管理过程中存在的风险，并且对存在的风险建立有效的规避措施。

3.5 本章小结

本章主要讨论了在线客服的整体需求和主系统对后台管理子系统的具体需求分析，先阐述当前环境的背景，然后结合系统业务分析提出了系统的功能性需求和非功能性需求，明确了在线客服的实现目标和方向。

第四章 后台管理子系统设计

4.1 后台管理子系统架构概述

4.1.1 后台管理子系统架构设计目标

根据需求分析，可以发现后台管理子系统主要通过对知识库管理、聊天记录查询等功能来对主系统给予支持，所以分析得知子系统基本是基于信息管理系统模型。应用 MVC 模式，把系统划分为模型层、视图层、控制层三层模型，并且把它们划分到各自相对独立的单元，以保证数据持久化、业务逻辑和界面显示的有效分隔，实现数据、界面和业务的分离，使系统更具有可维护性。

由于使用者基本是通过浏览器来使用系统，所以我们后台子系统也以基于 B/S 架构的模式进行开发。这样能是我们的系统具有广泛适用性，相关人员只需要接入相应网络通过浏览器即可登陆系统使用。

总之，后台管理子系统要为主系统提供一个易于操作，科学管理，可维护性高的后台信息管理系统。

4.1.2 后台管理子系统模块划分

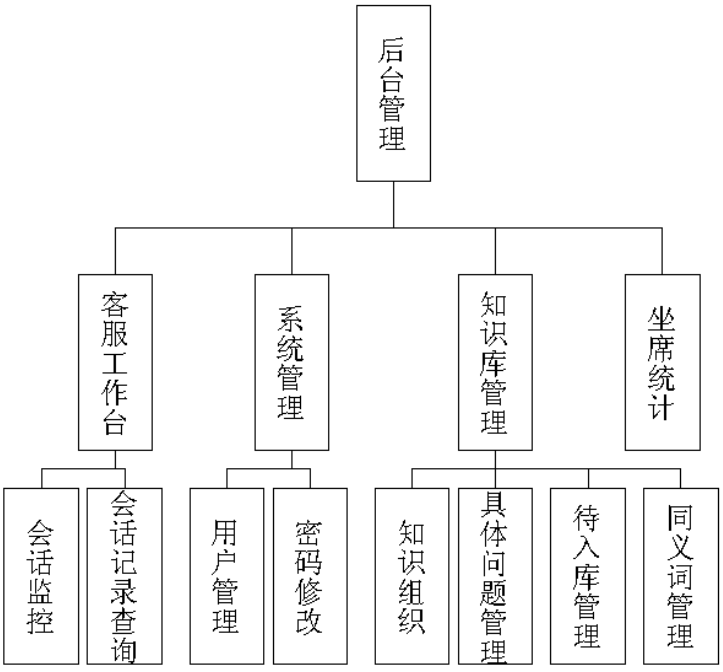

图 4-1 后台管理模块划分图

如图 4-1 ，后台管理子系统可以划分为以下 4 个模块：

(1) 客服工作台。包括会话监控和会话记录查询功能。

(2) 系统管理。包括用户管理及当前管理员密码修改功能。

(3) 知识库管理。包括知识组织，具体问题管理，待入库管理和同义词管理功能。

(4) 坐席统计。

各子模块功能描述在之前的需求分析中已有叙述，在此就不再详述了。

4.2 子系统数据库设计

后台子系统主要围绕聊天记录查询、知识库维护、客服人员资料维护这几个方面设计。所以只设计到整个系统中的一部分数据库表。

数据模型是现实世界中数据特征的抽象。数据模型应该满足三个方面的要求：

(1) 能够比较真实地模拟现实世界

(2) 容易为人所理解

(3) 便于计算机实现

在数据库的设计过程中，本文先从设计数据库概念数据模型开始，然后从数据库概念数据模型得出具体的物理数据模型[16, 17, 18, 19]，最后通过 PowerDesigner 生成具体的数据库生成 SQL 语句。

4.2.1 概念数据模型设计

概念数据模型(CDM)也称信息模型[20, 21, 22, 23]，它以实体－联系(Entity-RelationShip，简称E-R)理论为基础，并对这一理论进行了扩充。它从用户的观点出发对信息进行建模，主要用于数据库的概念级设计。

CDM 是一组严格定义的模型元素的集合，这些模型元素精确地描述了系统的静态特性、动态特性以及完整性约束条件等，其中包括了数据结构、数据操作和完整性约束三部分。

(1) 数据结构表达为实体和属性；

(2) 数据操作表达为实体中的记录的插入、删除、修改、查询等操作；

(3) 完整性约束表达为数据的自身完整性约束（如数据类型、检查、规则等）和数据间的参照完整性约束（如联系、继承联系等）；

本后台子系统数据库概念数据模型图(CDM)如图 4-2 所示。

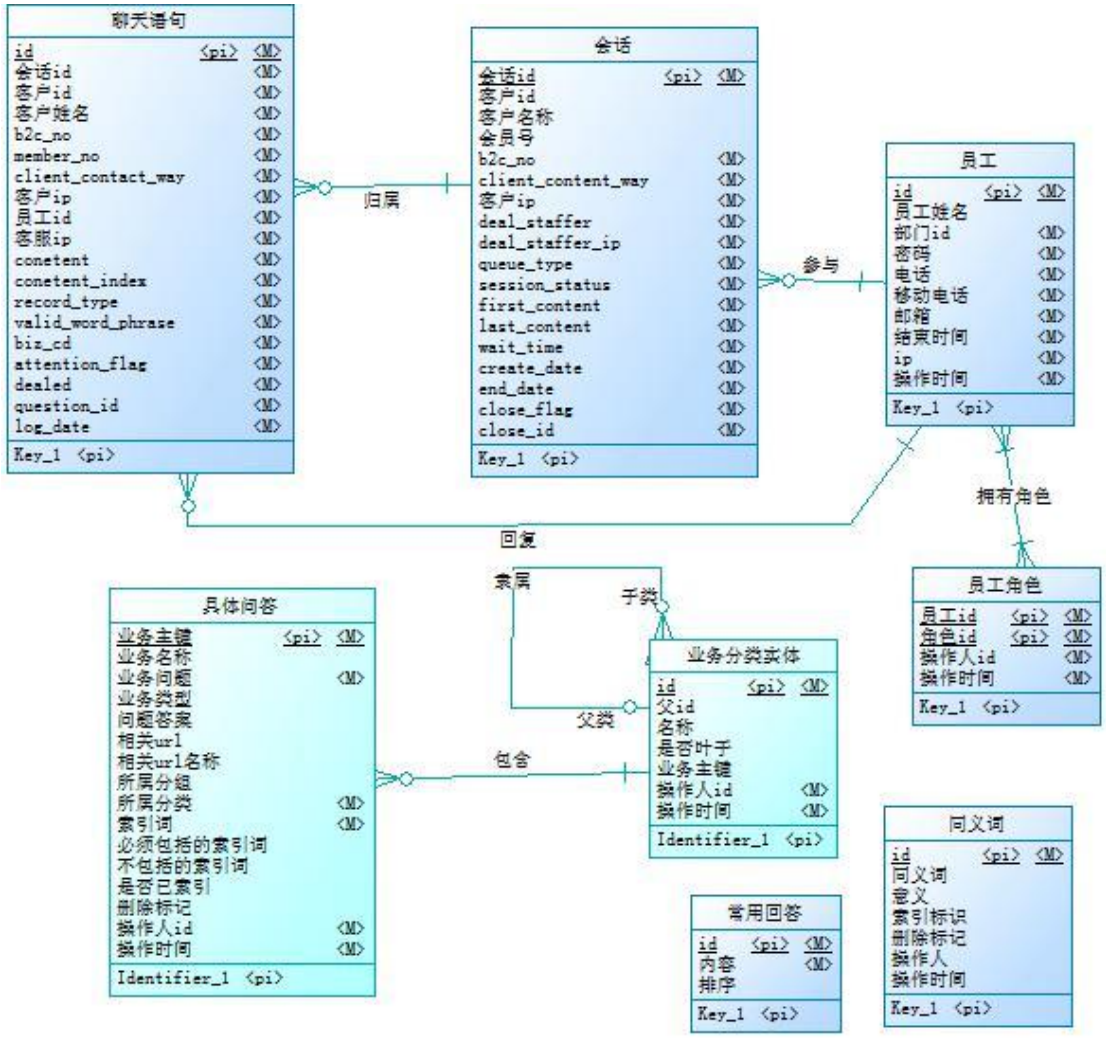

图 4-2 后台系统概念数据模型图（CDM）

在聊天记录查询方面，主要涉及了聊天语句表和会话表。每条聊天语句记录必定归属与某个会话记录，是多对一的关系。聊天记录表记录了具体的聊天内容，有客户的聊天记录、机器人的回答记录和客服的回答记录也记录在其中。会话表记录了当前会话的相关信息，如会话的开始、结束时间，会话类型等。聊天记录和会话表都与存放客服人员信息的员工表相关联。

在知识库维护方面，主要涉及具体问答表和业务分类表。具体问答表记录了所有的知识库中的问题与答案，以及其问题的关键词等其他辅助信息。业务分类表记录了知识库架构的分类树形数据，表呈递归模型结构，每条分类记录与其父分类关联到一起。每个业务分类都包含相应的问答，每个问答要属于某一个分类下。具体问题表与业务分类表呈多对一关系。目前初步把分类表树形的层数设置为三级，且具体问答只能属于三级分类。

在客服人员信息维护上面，涉及了员工表和员工角色表。员工表与员工角色

表是多对多关系。员工表记录了客服人员的详细人员信息，员工角色表记录了对应员工的角色。员工角色表以与员工 id 和角色 id 作为联合主键来维持与员工表的多对多关系。

在具体的物理数据表的设计上，还对聊天记录表和会话表设计了对应的归档表。由于所有的聊天记录和会话都记录在对应的 2 张表上，当数据量上升到一定程度的时候，那对数据的增删操作和检索的性能将会带来一定影响。所以设置了归档表来解决这个问题。原有的非归档表将对实时的聊天信息进行记录和检索，当聊天会话结束时系统将自动把非归档表的数据转移到归档表中。对历史记录的查询将只查询归档表。这样可以提升实时聊天时的响应速度。

另外后台管理还涉及了常用问答表和同义词表的维护。常用问答表是供客服人员能快捷的回复一些常用的回答，同义词表是供问题关键词中的同义的单词的映射的。这两个是相对独立的表，因为比较简单就不详述了。

4.2.2 物理数据模型设计

通常人们先将现实世界抽象为概念世界，然后再将概念世界转为机器世界。换句话说，就是先将现实世界中的客观对象抽象为实体(Entity)和联系(Relationship)，它并不依赖于具体的计算机系统或某个 DBMS 系统，这种模型就是我们上述所说的 CDM——概念数据模型，然后再将 CDM 转换为计算机上某个 DBMS 所支持的数据模型，这样的模型就是物理数据模型，即 PDM。

得出 PDM，也就得出了具体需要创建的数据库表中的各表的关系，各字段的定义，这样，就可以编写创建数据库表的 SQL 语句了。

数据库物理数据模型图(PDM)如图 4-3 所示。

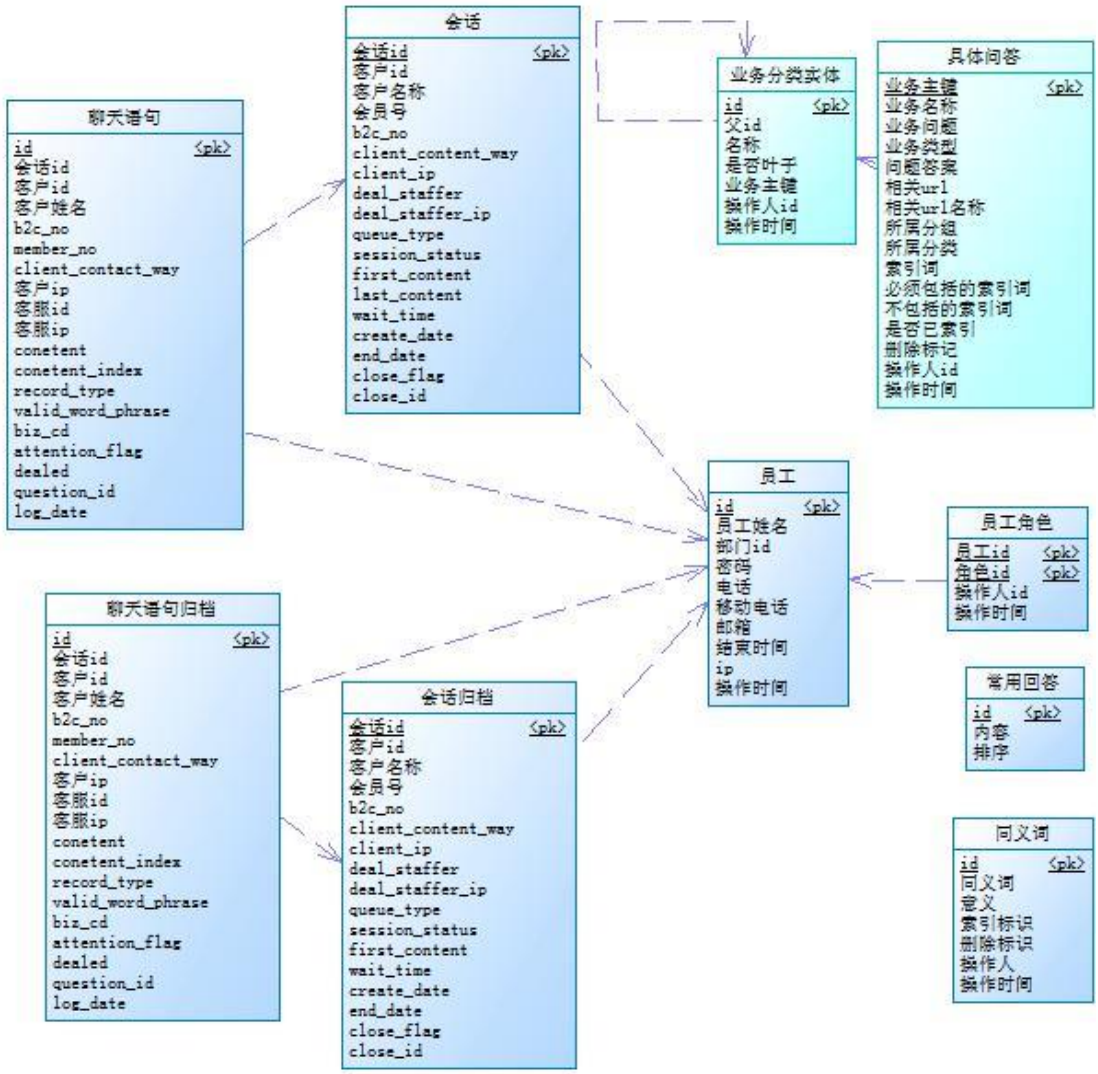

图 4-3 后台系统物理数据模型图（PDM）

4.3 总体类体系结构设计

科学合理的系统设计有利于提高系统的可维护性和可复用性。一个好的软件设计，必须能够允许新的设计要求以较为容易和平稳的方式加入到已有的系统中去，从而使这个系统能够不断地焕发出青春。一个可以重复使用的软件成分可以为将来的节省费用。一个构件被复用的频率越高，构件的初始开发投资就相对越少[7]。

本后台管理子系统总体上通过 MVC 分层架构[24, 25, 26, 27]，面向接口编程，使其满足“开-闭”原则(OCP)。“开-闭”原则是说在设计一个模块的时候，应当使这个模块在不修改的情况下被扩展，做到在不修改源码的情况下改变该模块的行为。这是面向对象设计的目标。

利用 Spring 框架的 IOC 容器来管理 Java Bean。在编程中使用 Java Bean 的接

口进行编程，在运行时通过 Spring 来提供具体的实现。使其满足依赖倒转原则(DIP)。DIP 原则主要就是讲要依赖抽象，不要依赖具体。传统的过程性系统的设计办法倾向于使高层次的模块依赖于低层次的模块，抽象层依赖于具体实现。DIP 原则就是要把这个错误的依赖关系倒转过来，只依赖抽象，不依赖具体。这是面向对象设计的主要机制。

多利用合成和聚合关系来代替继承关系，使其满足合成/聚合复用原则(CARP)。这个原则是说要尽量使用合成/聚合关系，而不是继承关系来组织新对象。CARP 原则就是在一个新的对象里使用一些已有的对象，使之为新对象的一部分，新对象通过这些对象的委派达到复用已有功能的目的。这样做可以在运行时间内进行对象的动态复用，新对象可以动态地引用与成分对象类型相同的对象。

本系统的设计从可维护性和可复用性的角度出发，通过上述的一些基本原则为指导进行合理设计，使开发的系统具有更高的可用性。

本系统的设计使用了 UML 可视化建模语言对设计进行描述、可视化处理，适合与描述面向对象开发过程。

4.3.1 整体类体系设计

在整体的类体系设计中，本后台子系统采用了 Anemic Domain Model[8]模型，即贫血的领域模型。简单来说，就是 domain object 只有属性的 getter/setter 方法的纯数据类，所有的业务逻辑完全由 business object 来完成(又称 Transaction Script)，这种模型下的 domain object 被 Martin Fowler 称之为“贫血的 domain object”。

如图 4-4 所示，主要描述了操作 domain object 的 service 层和负责持久化的 dao 层的关系。

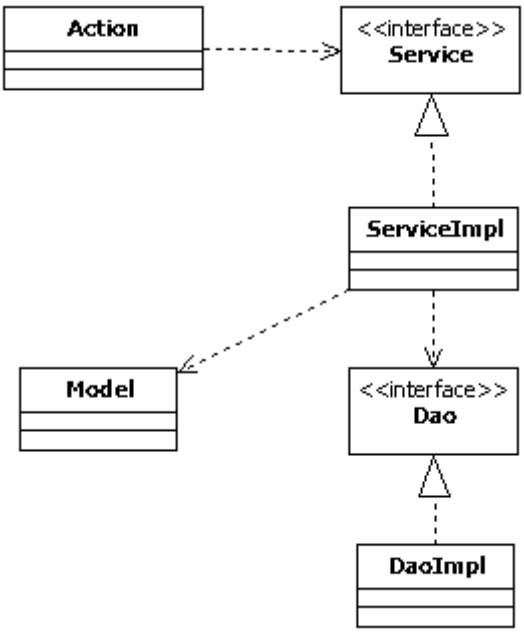

图 4-4 后台系统类体系示意图

Action 是表示层中的 Struts 的 Action 类，该类接收页面请求后调用服务层 Service 的接口。服务层的具体实现 ServiceImpl 则负责实际的业务逻辑，通过 Dao 对 Model 进行持久化操作。这里的 Model 是只有 getter/setter 方法的 Domain Object，主要充当一个数据容器。

这样设计可以把 Domain Object 的数据以及逻辑分割开来，是整体类体系易于设计，符合低耦合原则和单一职责原则，并且使得设计更趋稳定。此外，表现层与业务层，业务层与持久层之间的交互都是通过接口来调用的，这样满足 OCP 原则，对修改关闭，对扩展开放。而接口的实现由 Spring 容器来管理，通过依赖注入来获取具体的实现类。

4.3.2 可重用复合条件查询类设计

经过需求分析，本后台子系统具有较多的复合条件的列表分页查询。虽然不同的查询有不同的业务逻辑，但对于系统实现来说这些查询整体上都是一致的。它们有类似的查询条件，类似的分页需求，类似的数据库查询，所以可以通过设计一组可复用的查询类来完成符合条件分页查询的需求，以此来提高系统可复用性，提高开发效率。

如图 4-4 所示：

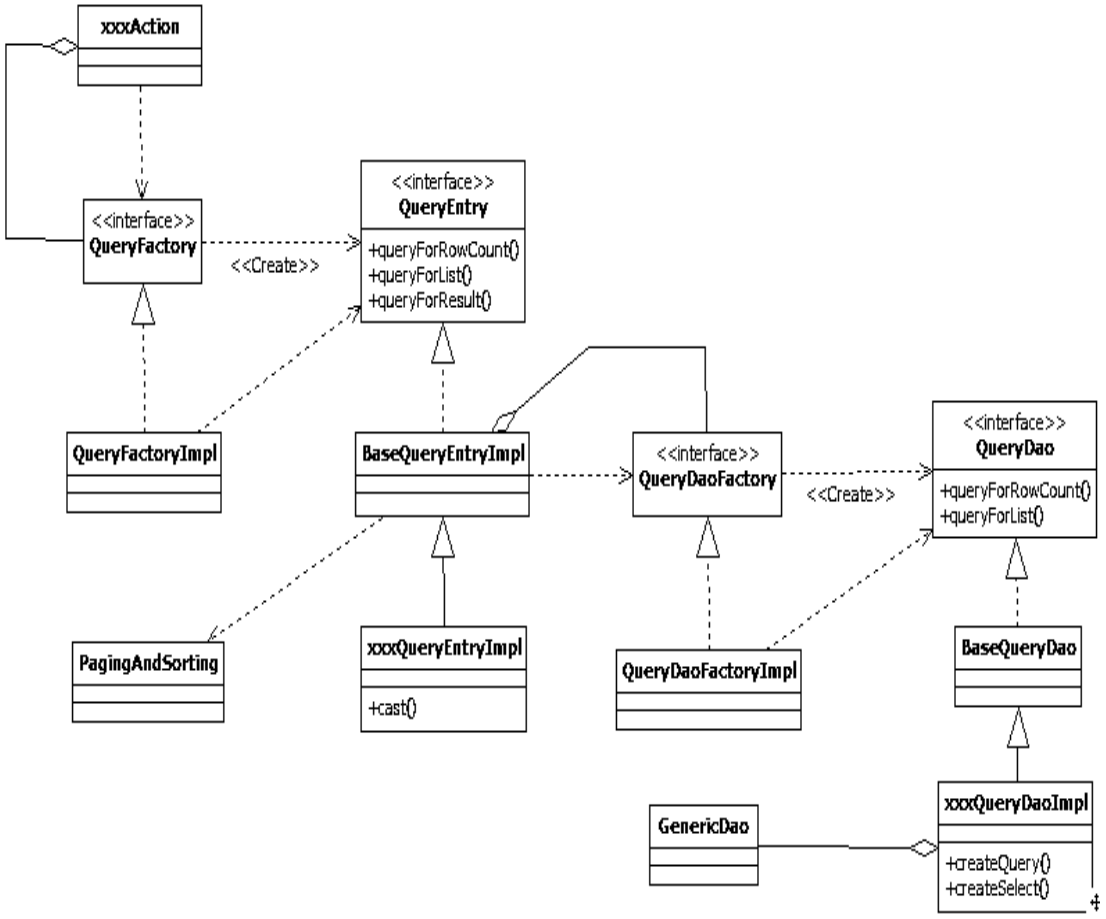

图 4-5 复合条件分页查询 UML 图

系统接收客户请求前，Spring 框架就把 BaseQueryEntryImpl 和 BaseQueryDao 的子类初始化并以 Map 方式保存为 QueryFactory 和 QueryDaoFactory 的成员。

首先，xxxAction 代表表现层，调用它本身一个成员 QueryFactory 的查询方法，QueryFactory 就会根据调用者调用 QueryEntry 的相应实现类来处理查询。

然后，QueryEntry 的实现类会进行页面的分页处理，并生成 PagingAndSorting 分页类，然后把具体查询交给 QueryDaoFactory 处理。

最后，QueryDaoFactory 调用 QueryDao 的相应实现类来执行具体数据库查询，返回的结果再在 QueryEntry 的实现类进行转换处理，最终返回给页面。

在图 4-4 中可以看到复合查询类的设计中，针对具体业务中各种不同的查询的实现类用工厂模式来管理，把共同的部分抽象为接口和抽象类，这样可以达到“开-闭”原则中对扩展开放，对修改关闭的要求。同时由于在查询处理里针对接口编程，各种不同业务的实现类都有共同的接口，所以可以通过抽象出来的方法进行统一处理，达到了里氏代换原则[28, 29]。另外，该部分的设计中还多出应用了聚合关系来组织类间关系，比如复合查询的 BaseQueryDao 的实现类和 GenericDao

类的关系就是聚合关系。这样做可以在代码运行是动态进行代码复用。

4.4 后台管理子系统界面设计

由于后台子系统面对的是内部人员，所以后台子系统的界面不需要太花哨。但是由于后台是履行管理职能，所以后台子系统的界面应该尽可能地做到简单易用，严谨，美观，大方。

如图 4-6 所示，后台子系统的界面风格主要以框架网页为基础，分上、左和右三个子框架网页。上侧为工具栏，左侧为导航条，右侧为内容主体。加上弹出消息子窗口的风格，弹出窗口使用消息提示组件 ymPrompt 的框架。

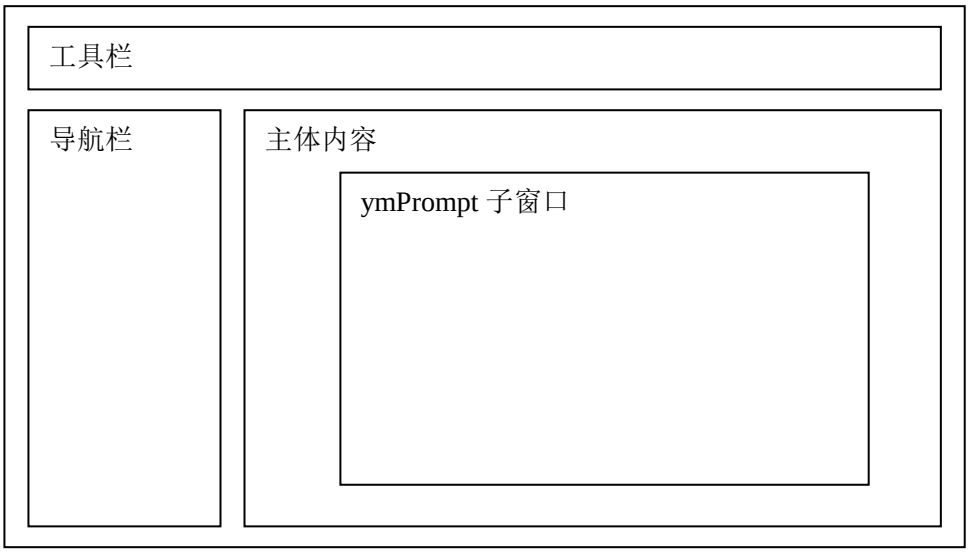

图 4-6 界面风格示意图

4.5 知识库模块设计

知识库模块主要包括知识组织，问题管理，待入库问题管理和同义词管理。

知识组织和问题管理在问题的修改上基本是一致，只是知识库查看的方式的出发点不同。知识组织是通过树形结构来一层层来选择浏览[30, 31]，而问题管理主要是通过符合条件查询来进行浏览的。所以在问题浏览上，对于知识组织主要通过设计知识库问题的树形结构，在问题管理上是应用可重用复合条件查询类组件来查询。对于知识组织和问题管理的知识库问题的增删改，则应用整体类体系的设计。

待入库问题管理本质上也是属于复合条件查询，不过由于条件固定，所以可以作为复合条件查询设计的简化版本来设计。

同义词管理也是属于复合条件查询的设计范畴，也有同义词的增删改，故可沿用知识库问题管理的设计方案。

知识库模块 UML 类图如图 4-7 所示：

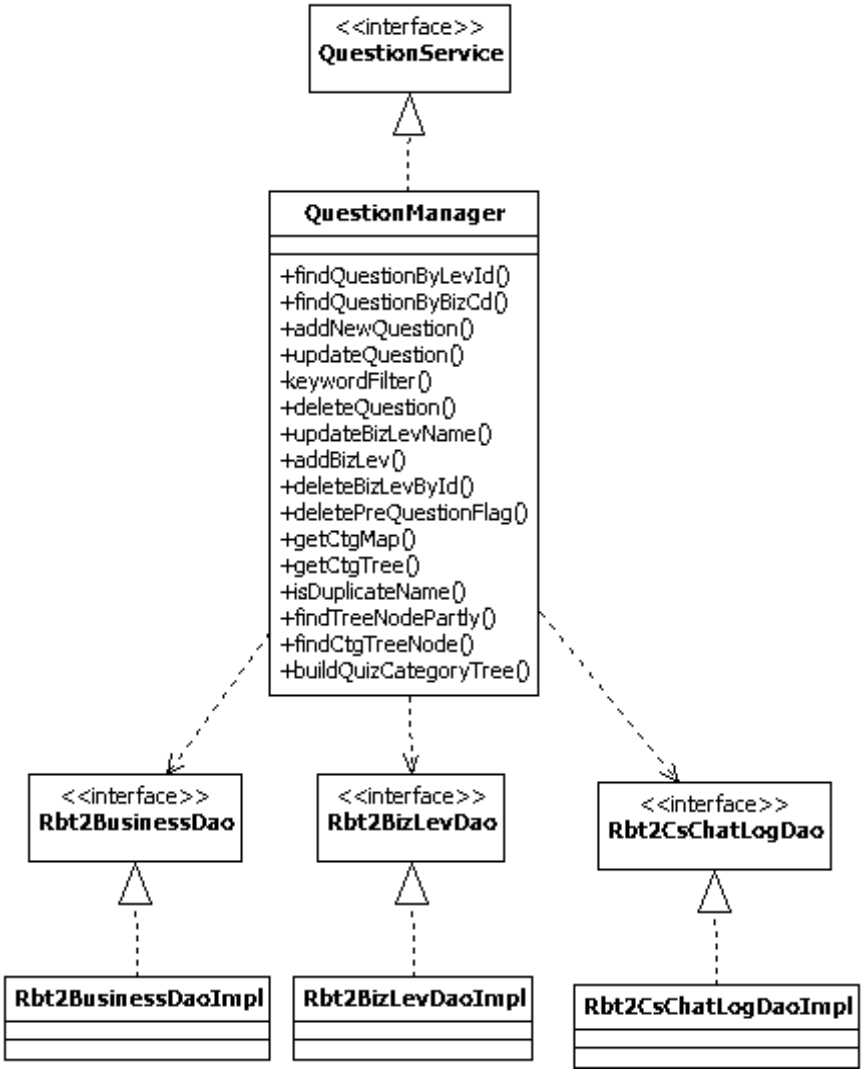

图 4-7 知识库模块类 UML 图

由图 4-7 看到，类 QuestionManager 的成员方法主要有对问题管理的增删改查、分类的增删改查和分类树形维护的相关方法。

分类树形维护中，采用了内存中保存分类树[32]的设计方法。即在第一次访问知识组织功能是，系统会先从数据库中读取分类并通过 buildQuizCategoryTree()方法来组织成树形结构，并保存在内存中。当下次再有访问的时候，就通过 getCtgTree()方法来从内存中获取树形结构而不是重新再次读取数据库。当分类树被修改时，系统将在下次读取分类树前重新读取数据库对分类树重新组织，更新内存中的分类树。这样可以有效提高获取分类数的效率，减少用户等待时间，缓解数据库压

力。

4.6 客服工作台模块设计

客服工作台包括聊天监控和历史记录查询。

聊天监控是对当前进行的会话进行查看并对会话进行操作，历史记录可以对所有会话进行查询。聊天监控和历史记录查询在会话的查询上均是对会话进行复合条件查询操作，所以设计上沿用了可重用复合条件查询查询类。

两者区别主要在流程上的不同处理。其 UML 活动图[33, 34]如图 4-8 与图 4-9 所示

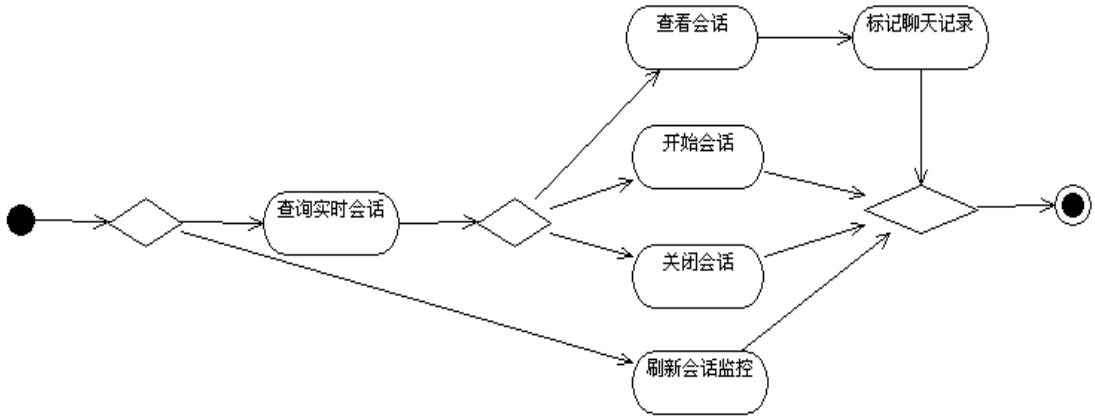

图 4-8 客服聊天监控活动图

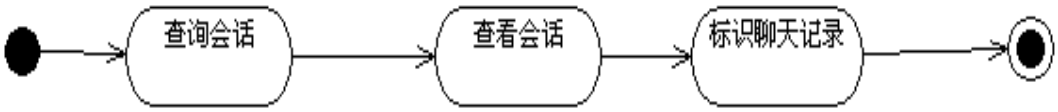

图 4-9 客服聊天记录查询活动图

系统当人工客服人手缺乏时可以切换到电子客服中，通过识别用户问题，利用后台知识库来对客户的问题进行简单的计算机自动应答。根据数据结构特点，选择队列技术实现。

(1) 首先，系统将该客户请求消息加入到会话队列中，供后续人工客服或电子客服获取；

(2) 当客服工作人员人手充足时，主动从会话队列获取会话，并开始与客户对话；

(3) 当客服工作人员人手缺乏时，机器人自动应答，由机器人来从队列中获

取会话回答：系统通过语义分析客户输入问题，把分析结果与知识库的索引文件相关联，搜索知识库中的问题答案，然后回答客户；

4.7 系统管理模块设计

系统管理模块最重要的就是对用户进行管理。

用户的管理主要包括用户的列表查询、用户资料维护和密码修改。用户的列表查询可以复用复合条件查询类的设计，其他功能可以沿用整体类体系的分层设计。其用户管理 UML 活动图如图 4-10 所示：

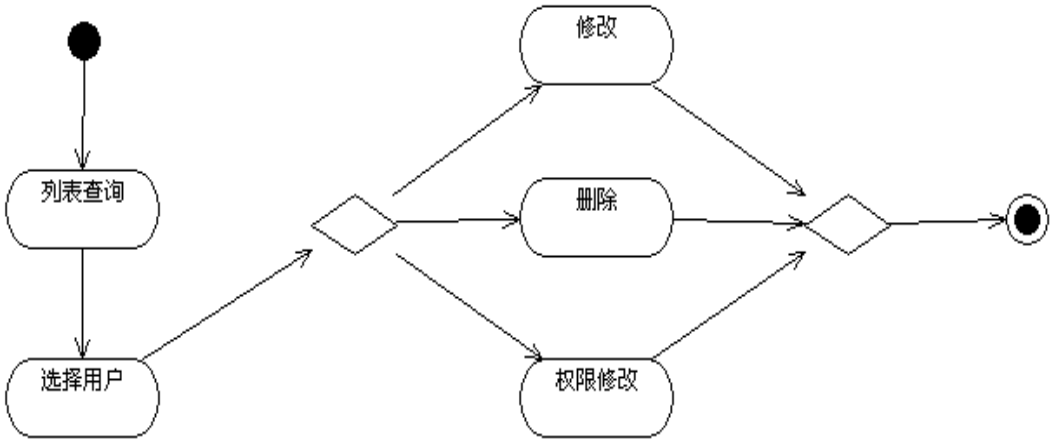

图 4-10 用户管理活动图

4.8 坐席统计模块设计

坐席统计模块就是对客服人员的工作量统计。

坐席统计可以通过时间来对所有的客服人员进行统计，系统会自动生成 excel 表格并提供下载。

生成 excel 表格的功能本系统应用了目前广泛成熟使用的 POI 框架。Apache POI 是 Apache 软件基金会的开放源码函式库，POI 提供 API 给 Java 程式对 Microsoft Office 格式档案读和写的功能[35, 36, 37]。本系统引用 POI 的 HSSF 函数库进行坐席统计的 excel 表生成。

坐席统计模块 UML 类图如图 4-11 所示：

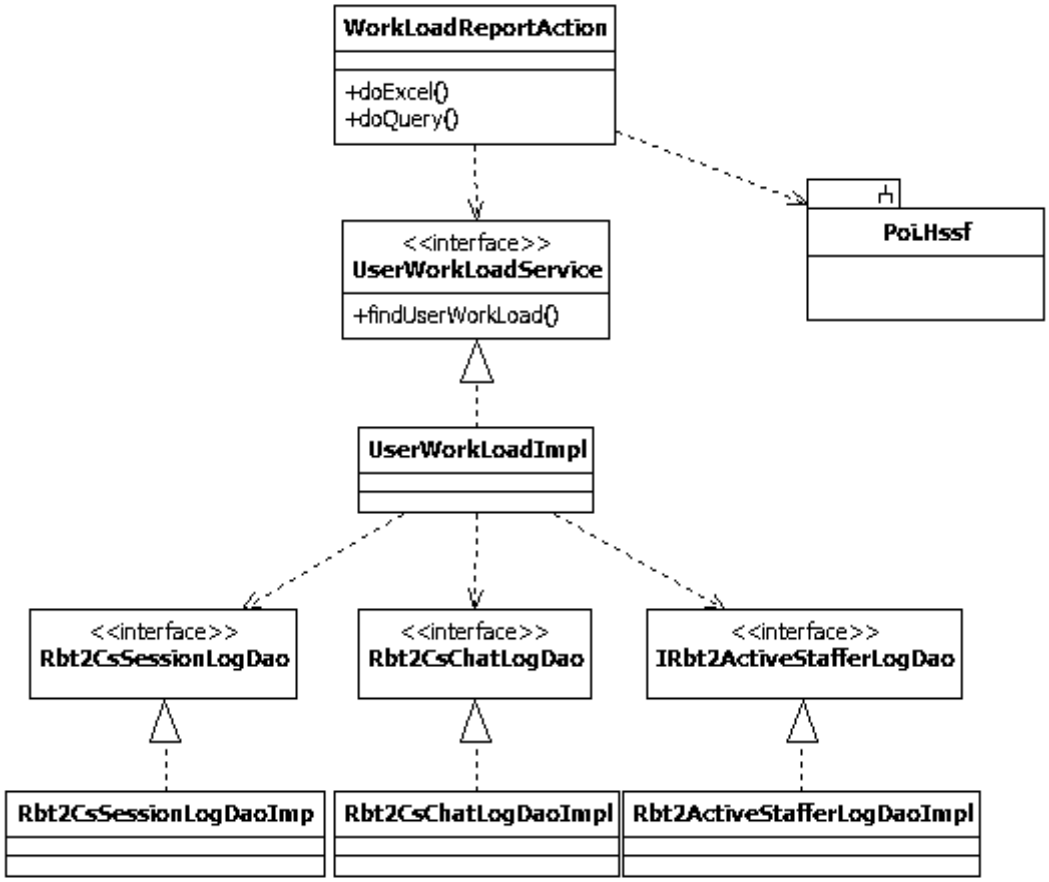

图 4-11 坐席统计类图

4.9 后台聊天平台设计

首先人工客服都是从队首取会话消息做回答。人工客服和机器人可以并行处理会话,，网站用户发起会话后，会话数据经过计算机解释后进入会话队列，遵循队列数据结构先进先出的特征，不论机器人还是处理完会话后标注进程结束后，才能取队列中下一队首会话。

算法描述：

第一步：定义数据结构 msgbuf，即消息；

第二步：生成消息队列；

第三步：服务器端启动消息队列，客户端注册后通信，实现读写操作，其中，switch 跳转语句控制客户端选择的是人工客服还是机器客服；若人工客服不在线，在线为 online，不在线为 outline，若 outline，则 switch 跳转自动退出，进入机器客服服务模式；

第四步：服务器段负责控制消息队列读取，并标志进程的开始和结束。

4.10 本章小结

本章主要讨论了后台管理子系统的设计问题。首先对子系统的数据库进行了设计，先从概念数据模型分析数据库，然后得出具体的物理数据模型，得出最终表设计。接着对系统整体类进行了设计，本系统应用了分层次的类体系的设计，应用了领域模型中的贫血模型。此外，对与频繁的复合条件查询的需求设计了一组可复用的查询类组件，以此来简化同类功能的开发，并且提高可维护性。最后再对每个模块进行了具体的设计。

本章说明了系统的详细设计，包括系统的概要设计，界面设计，数据库设计，总体类结构设计，各个功能模块设计等。下一章说明系统是如何实现的并在系统实现的基础上进行系统测试。

第五章 后台管理子系统的实现

5.1 知识库模块实现

5.1.1 知识组织实现

知识组织实现的关键在于分类树的数据结构设计，树型的构建和树形的序列化上。如图 5-1 所示。

图 5-1 知识库树形浏览示意图

(1) 分类树数据结构通过一个 QuizCategory 作为节点类实现。

节点类递归地拥有其孩子的列表，即 children 成员。id 是节点数据库主键，parentId 是父节点 id，text 是分类名称，其余的成员在树形序列化中详述。

```
public class QuizCategory implements java.io.Serializable
{
    private static final long serialVersionUID = 6707671508754920406L;

    private String id;
    private String parentId;
    private String text;
    private boolean hasChildren=false;
    private boolean expanded=false;
```

```
    private String classes;
    private List<QuizCategory> children;

    (其余方法为getter/setter…)
}
```

(2) 树形的构建主要通过 QuziManager 类中的 buildQuizCategoryTree()实现。

buildQuizCategoryTree()方法通过一次性遍历所有的分类节点就根据他们的父子关系建立了一棵分类树。并且把从数据库中生成的树保存在 QuizCtgCacheTree 类中的静态成员 ctgMap 中，以便下次获取能从内存中直接获取。

```
public QuizCategory buildQuizCategoryTree() throws Exception{
    try {
        List list = rbt2BizLevDao.findAllBiz();
        Map<String, QuizCategory> tree = new HashMap<String, QuizCategory>();
        String root = null;
        for (Object i : list) {
            Rbt2BizLev nodeDb = (Rbt2BizLev) i;
            QuizCategory node = castToQuizCategory(nodeDb);
            QuizCategory parentNode = null;
            if (root == null || node.getId().equals(root)) {
                root = node.getParentId();
            }
            if (!tree.containsKey(node.getParentId())) {
                // 在tree中初始化父结点
                parentNode = new QuizCategory();
                parentNode.setId(node.getParentId());
                tree.put(parentNode.getId(), parentNode);
            } else {
                // 在tree中获得父结点
                parentNode = tree.get(node.getParentId());
            }
            if (tree.containsKey(node.getId())) {
```

```
                // 如果之前曾被作为父节点添加入树则直接赋值而不入树
                ctgCopy(tree.get(node.getId()), node);
            } else {
                tree.put(node.getId(), node);
            }
            if (!parentNode.getId().equals(node.getId())) {
                parentNode.getChildren().add(tree.get(node.getId()));
            }
        }
        QuizCategory treeRoot = (QuizCategory) tree.get(root);
        //依据父子id是否相同判断是否合法根结点，不是则重建根结点和树
        if (treeRoot == null || !treeRoot.getId().equals(treeRoot.getParentId())){
            //重建根结点和数结构代码，在此略去
        }
        QuizCtgCacheTree.setCtgMap(tree);
        QuizCtgCacheTree.ROOT_ID = treeRoot.getId();
        return treeRoot;
    } catch (Exception e) {
        e.printStackTrace();
        LOG.error("构建分类树出错：" + e.getMessage());
        throw e;
    }
}
```

(3) 树形的序列化本系统使用 JSON 序列化来把分类树传输给页面。

利用 struts2 的 jsonplugin 类库的支持，QuizCategory 类的对象将会被序列化为 JSON 对象。在页面层中的树形列表采用的是 jQuery 的 treeView 类，这样只要把分类树的 JSON 数据传给页面的 treeView 就可以实现分类树形的展示了。

```
<script type="text/javascript">
……
$(document).ready(function(){
        $("#black").treeview({     url: "<s:property value='ShowTree' />" });
```

```
    });
……
</script>
……
<ul id="black">
</ul>
……
```

树的展示就是通过 Javascript 代码中 treeview()方法展示，jQuery 可以根据 Html 标签”black”来决定树形展现的位置。

(4) 知识组织实现截图。

图 5-2 展示了知识组织的具体页面图。

图 5-2 知识组织实现截图

```
public String input() throws Exception {
                performancesService.initPerformanceEditAction(perId, this);
        return SUCCESS;
    }

    @Override
    public String editSave() throws Exception {
```

```
            Boolean bln = performancesService.savePerformance(employeeId, perId,
                    perType, trParams,trParamsDel, getRequest());
            if (bln)
                return super.editSave();
            else
                return "error";
        }
        private PerformancesService performancesService;

        public void setPerformancesService(PerformancesService
    performancesService) {
            this.performancesService = performancesService;
        }
```

5.1.2 问题管理实现

问题管理实现的关键在于复合条件查询类组件的实现上。

复合条件查询类组件主要由 service 层的 QueryEntry、QueryFactory、BaseQueryEntryImpl 及其子类，dao 层的 QueryDao、QueryDaoFactory、BaseQueryDao 及其子类共同实现。

QueryEntry 接口实现：

```
public interface QueryEntry {
    Number queryForRowCount(Object _queryCondition);
    List queryForList(Object _queryCondition, PageBean pageBean);
    QueryResult queryForResult(Object _queryCondition, PageBean pageBean);
}
```

QueryFactory 接口实现：

```
public interface QueryFactory {
    Number queryForRowCount(String _beanId, Object _queryCondition);
    List queryForList(String _beanId, Object _queryCondition, PageBean pageBean);
    QueryResult queryForResult(String _beanId, Object _queryCondition, PageBean
pageBean);
```

```
}
```

BaseQueryEntryImpl 负责调用 dao 层查询的代码，其中 getQueryDao 获取得到的是具体的 dao 的名称，是根据当前实际的 QueryEntry 实现类的类名来构建的。

```
private List queryForList(Map queryCondition,PagingAndSorting pagingAndSorting) {
    List list = queryDaoFactory.queryForList(getQueryDao(), queryCondition,
    pagingAndSorting);
    Iterator it = list.iterator();
    List _list = new ArrayList();
    while (it.hasNext()) {
        _list.add(this.cast(it.next()));
    }
    return _list;
}
```

QueryDaoFactory 运行原理与 QueryFactory 类似，所以不再赘述。

经过了 service 层后，已经可以确定查询条件和需要调用具体查询的 dao 类的名称了，所以 dao 层的 QueryDaoFactory 根据名称获取到具体 QueryDao 的实例后，实现类可以调用父类 BaseQueryDao 的查询类执行数据库查询，具体的 sql 语句有具体实现类来覆盖父类。

BaseQueryDao 执行查询的代码片段：

```
public List queryForList(Map queryCondition,PagingAndSorting pagingAndSorting) {
    long begin = System.currentTimeMillis();
    Query _query = createQuery(queryCondition);
    List list = getGenericDao().find(createSelect(_query.getQueryString()),
_query.getNamedParameters(), pagingAndSorting);
    long end =System.currentTimeMillis();
    boolean debug = false;
    if(debug){
        System.out.println("queryForList:"+(end-begin));
    }
    return list;
}
```

QueryDao 的具体实例 QuestionQueryDaoImpl 的代码组建 SQL 片段：

```
protected Query createQuery(Map map) {
    ……
    StringBuffer buf = new StringBuffer();
    buf.append(" from " + Rbt2Business.REF + " a " );
    buf.append(" where a." + Rbt2Business.PROP_BIZ_TYPE + " = '" +
Rbt2Business.BIZTYPE_INFO + "' ");
    buf.append(" and a." + Rbt2Business.PROP_DELETED + " = 'N' ");
    buf.append(" /~ and a." + Rbt2Business.PROP_LEV_ID + " = {quizCategory} ~/");
    if (contentList!=null){
        buf.append(" and (1=0 ");
        for (String i : contentList){
            buf.append(" or a." + Rbt2Business.PROP_INDEX_WORDS + " like '%" +
i + "%' " );
        }
        buf.append(" ) ");
    }
    buf.append(" order by a." + Rbt2Business.PROP_LEV_ID );
    Query query = QueryUtils.filterQuery(buf.toString(), map);
    return query;
}
```

问题管理的实现截图如图 5-3 所示：

图 5-3 问题管理实现截图

```
public class FindCurrentProcesshandleStepAction extends PageBaseAction {
    private static final long serialVersionUID =
-8317809282598348414L;
    // 此参数 为接受参数 值为 以上 所列
    public String step = "1";
    @Override
    public String input() throws Exception {
        return SUCCESS;
    }
    public String getStep() {
        return step;
    }
    public void setStep(String step) {
        this.step = step;
    }
```

5.1.3 同义词管理实现

同义词管理沿用了复合查询组件[38]的设计，故不再赘述。

同义词管理实现界面截图如图 5-4 所示：

图 5-4 同义词查询截图

```
public String input() throws Exception {
        this.setStartDate(DateTime.convertDateTime(new Date(), "yyyy") + "01");
        this.setEndDate(DateTime.convertDateTime(new Date(), "yyyy") + "12");
        resultDeptTypeList =
departmentsService.findDepartmentByAccountId(this.getAccountIdFromSession())
;
        groups = groupService.findAllGroups(getDepartmentIdFromSession(),"");
        return SUCCESS;
    }
    //统计同义词
    public String searchList() throws Exception {
        this.setPageCount(10);
        resultData = performancesService.findStaRePortFroms(
                groupId
                ,this.convertToUTF8(deptName).trim()
                ,this.convertToUTF8(accountName)
                ,startDate
                ,endDate
                , this);
        return SUCCESS;
    }
    //搜索 同义词
```

```
    public String inputSearchByPerentIdList() throws Exception {
        this.setStartDate_(DateTime.convertDateTime(new Date(), "yyyy") +
"01");
        this.setEndDate_(DateTime.convertDateTime(new Date(), "yyyy") +
"01");
        resultDeptTypeList =
departmentsService.findDepartmentByAccountId(this.getAccountIdFromSession())
;
        return SUCCESS;
    }
    //搜索 同义词
    public String searchByPerentId() throws Exception {
            resultData = processhandleService.getSearchByPerentId(
                    this.getAccountIdFromSession()
                   ,this.convertToUTF8(accountName_)
                   ,deptName_
                   ,this.convertToUTF8(startDate_)
                   ,this.convertToUTF8(endDate_)
                   ,this);

        return SUCCESS;
    }
```

5.1.4 待入库问题管理实现

待入库问题管理对复合查询组件做了简化处理，即对输入条件置空：

```
public void doList() {
    getQueryParam().setQuery(true);
    setQueryResult(queryFactory.queryForResult("preQuestionQueryEntry",new
HashMap(), getQueryParam().getPageBean()));
}
```

其中函数参数的 new HashMap()原本是传入输入条件的 JavaBean，但在待入库

问题中只需要传入一个空的 Map 对象即可。这样待入库问题就直接根据底层 dao 中的 SQL 语句直接查询[39]。

待入库问题管理实现界面截图如图 5-5 所示：

图 5-5 待入库问题查询截图

```
private static final long serialVersionUID = 2660427464423040426L;
    private ArrayList<ArrayList<String>> resultData;
    private String accountId;
    private String startDate;
    private String endDate;
    private String deptId;

    private EmpdeptjobHistoryService empdeptjobHistoryService;

    public String input() throws Exception {
        System.out.println(deptId);
        return SUCCESS;
    }

    public String searchList() throws Exception {
        startDate = this.convertToUTF8(startDate);
        endDate = this.convertToUTF8(endDate);

        resultData = empdeptjobHistoryService.createPageForChange(accountId,
startDate, endDate, this);

        return SUCCESS;
    }
```

5.2 客服工作台模块实现

5.2.1 会话监控实现

会话监控功能的主界面截图如图 5-6 所示：

图 5-6 会话监控实现截图

```
protected void readAccountDetail() throws Exception {
        if (accountId == null || "".equals(accountId)) {
            throw new MsgException("参数错误，找不到需要的accountId参数，
请重试！");
        }
        tsAccountService.findDataForDetail(accountId, this);
    }
protected String packDataToTree(ArrayList<TreeBean> treeBeanList) {
        final String customBasePath = this.getBasePath();

        UserDataUncoder orgUncoder = new UserDataUncoder(){
            public Object getID(Object pUserData) throws UncodeException {
                TreeBean treeBean = (TreeBean)pUserData;
                return treeBean.getId();
            }
    public Object getParentID(Object pUserData) throws UncodeException {
                TreeBean treeBean = (TreeBean)pUserData;
```

```
                return treeBean.getParentId();
            }
        };

        AbstractWebTreeModelCreator treeModelCreator =
         new AbstractWebTreeModelCreator(){
    protected Node createNode(Object pUserData, UserDataUncoder pUncoder) {
                TreeBean treeBean = (TreeBean)pUserData;
                WebTreeNode webTreeNode = new
WebTreeNode(treeBean.getName(), "mid" + treeBean.getId());

                webTreeNode.setSelected(treeBean.isSelected());
                webTreeNode.setAction("javascript:void(0);");
                webTreeNode.setValue(treeBean.getId());

    webTreeNode.setIcon(customBasePath+"image/common/treebutton/showNor
mal.gif");

    webTreeNode.setOpenIcon(customBasePath+"image/common/treebutton/show
Open.gif");
                return webTreeNode;
            }
        };
public String editSave() throws Exception {
        if (accountId == null || "".equals(accountId)) {
            throw new MsgException("接收参数错误，没有接受到参数
accountId !");
        }
        if (newauthorparticularIds == null || "".equals(newauthorparticularIds)) {
            throw new MsgException("请先选择预授予权限！ ");
        }
        tsUrlMapService.addAuthorParticular(accountId, newauthorparticularIds);
```

```
        return SUCCESS;
    }
```

5.2.2 会话记录查询实现

会话记录查询功能的主界面截图如图 5-7 所示：

图 5-7 会话记录查询截图

会话记录详细查看截图如图 5-8 所示：

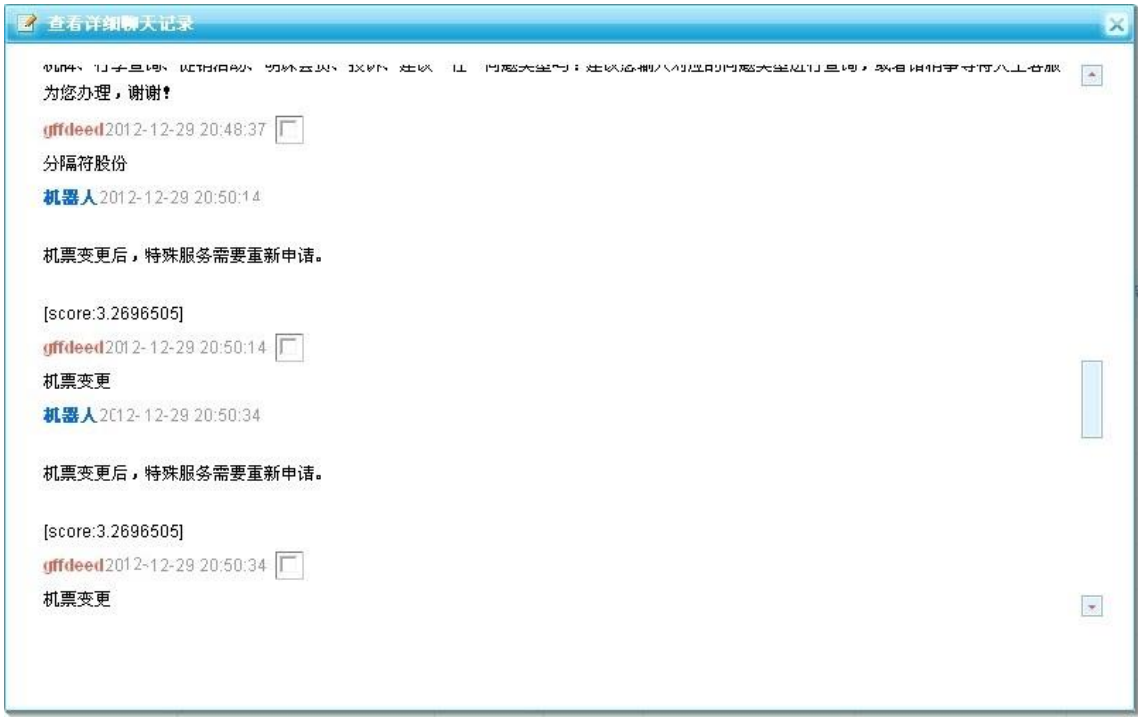

图 5-8 会话记录查询截图

```
public String input() throws Exception {
        this.setStartDate(DateTime.convertDateTime(new Date(), "yyyy") + "01");
        this.setEndDate(DateTime.convertDateTime(new Date(), "yyyy") + "12");
        resultDeptTypeList =
departmentsService.findDepartmentByAccountId(this.getAccountIdFromSession())
;
        groups = groupService.findAllGroups(getDepartmentIdFromSession(),"");
        return SUCCESS;
    }
    public String searchList() throws Exception {
        this.setPageCount(10);
        resultData = performancesService.findStaRePortFroms(
                    groupId
                    ,this.convertToUTF8(deptName).trim()
                    ,this.convertToUTF8(accountName)
                    ,startDate
                    ,endDate
```

```
                , this);
        return SUCCESS;
    }
    public String inputSearchByPerentIdList() throws Exception {
        this.setStartDate_(DateTime.convertDateTime(new Date(), "yyyy") +
"01");
        this.setEndDate_(DateTime.convertDateTime(new Date(), "yyyy") +
"01");
        resultDeptTypeList =
departmentsService.findDepartmentByAccountId(this.getAccountIdFromSession())
;
        return SUCCESS;
    }
    public String searchByPerentId() throws Exception {
            resultData = processhandleService.getSearchByPerentId(
                    this.getAccountIdFromSession()
                   ,this.convertToUTF8(accountName_)
                   ,deptName_
                   ,this.convertToUTF8(startDate_)
                   ,this.convertToUTF8(endDate_)
                   ,this);

        return SUCCESS;
    }
```

5.2.3 后台聊天实现

系统当人工客服人手缺乏时可以切换到电子客服中，通过识别用户问题，利用后台知识库来对客户的问题进行简单的计算机自动应答。根据数据结构特点，后台聊天实现如下：

服务器程序 ChatServer 负责监听客户端的请求，记录当前的在线用户列表，并将用户发送的消息推送给在线客服，而客户程序 ChatClient 负责向服务器发送消息，并接受由服务器中转的由机器人或者人工客服发送给自己的消息。

客户端是一个 Windows 窗体应用程序[40, 41, 42, 43, 44]，首先创建一个 Windows 窗体应用程序；在服务器端使用了多线程，每个用户通过一个单独的线程进行连接，当聊天服务器开始运行时，它就启动一个线程等待客户连接（在方法 StartListen（）中实现）。当接收到一个请求时，服务器立刻启动一个新的线程来处理和该用户端的信息交互（在方法 ServerClient（）中实现）。这里自定义了一个 Client 类，它用于保存每个当前在线用户的用户名和与服务器连接 Socket 对象。当 Socket 连接一旦建立，就马上将其保存在一个 Client 对象中，以便让每个用户有自己的 Socket，以后可以对不同用户的 Socket 对象进行操作，实现与客户端的数据交流。

主要实现代码：

```
struct msgbuf{
    long mtype;
    int subtype;     //1->register 2->broadcast 3->unregister
    int pid;
    char nick_name[N];
    char mtext[N];
};
//定义消息正文大小的宏
#define MSG_LEN (sizeof(struct msgbuf) - sizeof(long))
int main(int argc,char *argv[])
{
    key_t key;
    int msgid;
    pid_t pid;
    int TYPE_ME;     //由 getpid()来指定唯一该进程才有的 mtype
    struct msgbuf msg;
    TYPE_ME = getpid(); //用自己进程的进程号作为自己收消息的消息类型
    //argv[1]用于接收昵称
    if(argc < 2){
        printf("Usage:%s <nick_name>\n",argv[0]);
        exit(-1);
```

```
}
//生成消息队列的 key
if((key = ftok("./",'s')) < 0){
    perror("ftok error.");
    exit(-1);
}

//如果服务器已创建消息队列才可打开，客户端无权创建
if((msgid = msgget(key,0666)) < 0){
    printf("[server maybe not on line...]\n");
    exit(-1);
}
//每启动一个客户端要先实现一个注册操作
msg.mtype = TYPE_SRV;
  msg.subtype = 1;
msg.pid = getpid();
strcpy(msg.nick_name,argv[1]);
msgsnd(msgid,&msg,MSG_LEN,0);    //MSG_LEN 是消息正文大小

//客户端双进程实现读和写操作
if((pid = fork()) < 0){
    perror("fork error.");
    exit(-1);
}
else if(pid == 0){ //子进程专门负责读消息
    while(1)
    {
        //msgid 为消息队列的队列 ID
        //&msg 为消息类型为 TYPE_ME 的消息缓冲区首地址
        //0 指定了阻塞式等待
        msgrcv(msgid,&msg,MSG_LEN,TYPE_ME,0);

        //判断接收到的消息是否为 quit
```

```
        if(strncmp(msg.mtext,"quit",4) == 0){
            printf("[server will close in 3 seconds...]\n");
            kill(getppid(),SIGUSR1);//若收到 quit 通知父进程并自己退出
            exit(0);
        }
        if(strncmp(msg.mtext,"exit",4) == 0){
            msg.subtype = 3;
        }

        //写成 switch 方便以后扩展
        switch(msg.subtype)
        {
        case 1 :
            printf("[%s on line]\n",msg.nick_name);
            break;
        case 2 :
            printf("[%s]\n",msg.nick_name);
            printf("%s",msg.mtext);
            break;
        case 3 :
            printf("[%s off line]",msg.nick_name);
            break;
        default :
            break;
        }
    }
}
else{   //父进程专门负责写消息
    msg.mtype = TYPE_SRV;   //设定服务器的发送类型
    msg.subtype = 2;    //设为广播模式
    while(1)
    {
        printf("[%s]\n",msg.nick_name);
```

```
            fgets(msg.mtext,N,stdin);
            msgsnd(msgid,&msg,MSG_LEN,0);
            //当此进程输入 exit 时，表示该进程退出聊天
            //由服务器通知其他进程，该进程已下线
            if(strncmp(msg.mtext,"exit",4) == 0){
                msg.subtype = 3;
                msgsnd(msgid,&msg,MSG_LEN,0);
                sleep(1);
                kill(pid,SIGKILL);
                exit(0);
            }
        }
    }
    return 0;
}
```

5.3 系统管理模块实现

用户信息列表查看截图如图 5-9 所示：

图 5-9 会话记录查询截图

```
public String input() throws Exception {

        jobOptionBeans = jobsService.findAll();
        deptOptionBeans = departmentsService.findDepartmentAll();
        accountOptionBeans = tsAccountService.findNewAccount();

        return SUCCESS;
    }

    public String addSave() throws Exception {

        empdeptjobService.add(parentempId, deptId, jobId, accountId, jobLevel);
        return SUCCESS;
    }
```

用户资料编辑截图如图 5-10 所示：

图 5-10 用户资料编辑截图

用户权限修改功能截图如图 5-11 所示：

图 5-11 用户权限修改截图

5.4 坐席统计模块实现

坐席统计模块实现截图如图 5-12 所示：

图 5-12 坐席统计模块截图

5.5 本章小结

本章主要展示了系统的实现效果，在线客服聊天后台系统的功能实现针对系统的需求进行系统功能模块的实现，在该系统需求分析和概要和详细设计的基础上进行进一步的功能实现和系统编程实现工作本章对知识库模块进行了详细的实现的说明，阐述了系统中的核心部分，然后再介绍其余各部分的实现效果。

第六章 系统测试

6.1 后台管理子系统的测试

6.1.1 测试目标

为保证软件质量，使软件上线后正常运行，必须对软件进行充分的测试[45]。

测试目标就是提高软件的质量。

软件测试贯穿系统设计的整个生命周期过程[30]，从单元测试到功能测试，再到整体性能测试。。

6.1.2 测试范围

测试范围主要涉及后台管理子系统的日常工作。所以测试范围包括知识库模块中的知识组织相关功能，问题管理的相关功能，待入库问题的管理，同义词管理，客服工作台模块中的会话监控相关功能，历史会话查询功能，系统管理模块中的用户管理功能最后是客服工作量统计模块中的工作量统计功能。本系统的测试工作将会进行数据持久化层的单元测试，系统集成测试，然后到系统综合测试，还有修改完 bug 后的回归测试等工作。

测试对后台聊天系统的正常运行起至关重要的作用。本后台聊天管理系统必须具备可靠、可扩展、可维护等特性。

6.1.3 测试环境

测试环境包括硬件环境和软件环境，本软件测试环境如下：

硬件环境：内存：2G、 CPU：双核，2.26Ghz

软件环境：客户端：浏览器 IE8.0；

数据库：ORACLE；

操作系统：Windows XP 以及以上。

6.1.4 测试用例

测试用例如表 6-1 所示：

表 6-1 系统测试用例

编号	模块名	操作步骤
1	知识库模块	对业务类型进行增删改
2		对问题分类增删改；对具体问题进行增删改。包含问题标题，内容和关键词和相应的 url
3		查看已经处理待入库问题，可选删除按钮删除待入库问题
4		对知识库进行查看，可通过树形目录。
5		点击“上传”按钮，可查看系统提示。
9	客服工作台模块	查询会话，监控和管理会话以及查看会话的详细信息，客服人员可以对待处理的会话进行应答回复以启动人工客服。
10		查看过去结束了的会话历史记录
		点击‘回复’，可以对待处理的回话进行回复应答以启动人工服务。
11		查询的会话记录包括电子客服或者是人工客服
12	坐席管理模块	选择某时间段，点击，可以查看某时间段某客服人员的工作量。
13		点击“导出”，系统会自动生成 excel 表格并可提供下载。
25	系统管理模块	用户的管理模块可查询用户列表，并对已有用户增删改，对用户资料维护。
26		密码修改。
27		用户权限修改。

6.2 测试结果

系统实际结果如表 6-2 所示：

表 6-2 系统测试结果

编号	模块名	预期测试结果
1	知识库模块	知识组织，问题管理，待入库问题管理和同义词管理
2		对问题分类进行增删改。
3		对具体问题进行增删改。包含问题标题，内容和关键词和相应的 url
4		通过树形目录对知识库的组织结构进行查看。

5		点击“上传”按钮，可查看系统提示。
6	客服工作台模块	查询当前进行中的实时会话， 会话监控及会话记录查询。客服人员在系统中实现对各种状态的会话进行监控和管理。客服可以查询当前进行中的实时会话或者过去结束了的会话历史记录
7		浏览会话历史记录
8		点击‘回复’，可以对待处理的回话进行回复应答以启动人工服务。
9		查询的会话记录包括电子客服或者是人工客服
10	坐席管理模块	选择某时间段，点击，可以查看某时间段某客服人员的工作量。。
11		点击“导出”，系统会自动生成 excel 表格并可提供下载。
12	系统管理模块	查询用户列表，并对已有用户增删改，对用户资料维护。
13		密码修改。
14		用户权限修改。

6.3 本章小结

本章是系统的测试工作，对后台管理子系统的测试工作做了整体上的说明，针对系统需要用到功能进行的测试，给出了系统的测试方法及结论。包括测试目标，测试范围，测试环境和测试用例。通过测试，给出系统测试结果。

第七章 总结和展望

7.1 全文小结

互联网就是当前发展最为迅猛的一种知识分享、交流工具，而在科技迅猛发展的同时，企业之间的竞争也越来越激烈。企业之间的竞争不但表现在产品上，还表现在服务上。所以很多企业目前都借助互联网技术，纷纷退出电子客服系统来简化客服工作，同时提高与客户沟通的效率，一举两得。本文从实际出发，结合当前的市场需求提出了一种即可提供人工服务，又可提供机器人服务的新型互联网在线聊天客服系统。本文主要讨论的是该客服聊天系统的后台子系统。

本文主要完成了如下工作：

(1) 本文详细分析了当前形势下多功能在线客服系统出现的需要，并且深入讨论了该客服系统所依赖的后台子系统的功能需求。

(2) 结合时代背景分析了实现这样一个后台管理子系统所需要的技术，以及对后台管理子系统进行了一个科学合理的架构分析，并且给出了一个总体架构。在这个总体架构的指导下，对后台管理子系统进行了详细设计，明确了后台管理子系统的各功能的实现逻辑。

(3) 在详细设计的指导下实现了一个支持提供人工服务和机器人服务的在线客服的后台管理子系统。该子系统可以为主系统提供当前聊天会话监控，历史聊天会话记录查询，知识库的查看，知识库问题的查询以及维护，用户管理和坐席工作量统计等功能。

(4) 对主要的类进行代码详述，并展示系统成果（界面展示）。对系统测试方案进行概述，包括系统单元测试，集成测试等。

7.2 展望未来

后台管理子系统是从整体上来讲是一个信息管理系统，它对各种聊天信息、问答信息、人员信息进行了增、删、改、查等操作。它对操作者进行了简单的权限控制。后台管理子系统只是完成了基本的功能，日后需要在多方面上进行完善和扩展。

(1) 在安全方面，需要进一步提高安全监控，对不同身份的人需要根据权限进行访问控制。对相关敏感信息需要经过加密处理。

(2) 在坐席统计方面，需扩展统计功能，比如在页面上也能查询统计信息。

(3) 在问题管理方面，需要增加问题关联功能，坐席在查看一个问题的同时系统应该能列出与此问题相关联的问题列表供坐席参考。

致谢

本论文从选题、搜集阅读和研究文献到写作和最后的定稿，都凝聚着导师无限的教诲。我最想发自内心地向我的导师表示感谢。

另外，我要感谢我的家人，特别是我的父母。他们对我无私的支持和鼓励，是我顺利完成学业的精神支柱。

最后，我要特别感谢我的同学和学长，他们在我论文写作的各个阶段，给予我很多建议和指导，论文的顺利完成离不开他们的精心指导。

参考文献

[1] Web2.0 特点纪要[EB/OL].
http://12oak.spaces.live.com/Blog/cns!1pGb53FsycpwCL2pwp4ikHkw!203.entry.

[2] 客户服务[EB/OL]. http://baike.baidu.com/view/535872.htm.

[3] 谢廷彦. 答疑系统中的分类问题研究[D]. 湖南大学硕士学位论文,2008.

[4] 王志臣, 王世锋. 基于 Struts 和 Hibernate 的多层构架在 JAVA Web 开发中的应用[J].计算机与数字工程,2007.

[5] 现代 Java web 开发架构分析[EB/OL]. http://www.builder.com.cn/2008/0425/833410.shtml

[6] 刘新福. Struts2 快速入门[EB/DK].

[7] 阎宏. Java 与模式[M]. 电子工业出版社,2002.

[8] Martin Fowler. Anemic Domain Model [EB/OL].
http://www.martinfowler.com/bliki/AnemicDomainModel.html

[9] 总结一下最近关于 domain object 以及相关的讨论[EB/OL].
http://www.javaeye.com/topic/11712

[10] Apache POI 维基百科 http://zh.wikipedia.org/zh-cn/Apache_POI [EB/OL].

[11] Rick,Stefan.. 系统的软件测试(Systematic Software Testing)[M]. 2002

[12] 刘湛 J2EE 全面介绍 http://www.ibm.com/developerworks/cn/java/j2ee/[EB/OL].

[13] AJAX 简介 http://www.w3school.com.cn/ajax/ajax_intro.asp [EB/OL].

[14] James Rumbaugh, Ivar Jacobson, Grady Booch. UML 参考手册第二版[M]. 机械工业出版社,2005.

[15] James Rumbaugh, Ivar Jacobson, Grady Booch.UML 用户指南第二版[M]. 人民邮电出版社,2006.

[16] 萨师煊,王珊. 数据库系统概论. 北京. 高等教育出版社. 2004.

[17] 陈信,沈建强. 计算机网络与网页制作. 上海. 上海交通大学出版社. 2003.

[18] 江卫东. 人力资源管理理论和方法. 北京. 经济管理出版社. 2002.

[19] 赵致格. 数据库系统与应用(SQL Server). 北京. 清华大学出版社. 2005.

[20] 丁宝康,董健全. 数据库实用教程. 北京. 清华大学出版社. 2001.

[21] 王晟,万科. 数据库开发经典案例解析. 北京. 清华大学出版社. 2005.

[22] 邓文渊. ASP 与网页数据库设计. 北京. 中国铁道出版社. 2001.

[23] 杨云. ASP.NET 典型系统开发详解. 北京. 人民邮电出版社. 2006.

[24] 石志国. ASP 动态网站编程. 北京. 清华大学出舨社. 2001.

[25] 李昆,叶炜,任刚. SQL Server 2000 课程设计案例精编. 北京. 中国水利水电出版社. 2005.

[26] J C. Ports，T. Giddens，B. Yadav Surya.The Development and Evaluation of an Improved

Genetic Algorithm Based on Migration and Artificial Selection[J]，IEEE Transactions On Systems，Man and Cybernetics,1994,24(1):73-86

[27] Holland J．H.Genetic Algorithms and Classifier Systems：Foundations and Future Directions，Genetic Algorithms and Their Applications，Proc of the 2nd Int On Genetic Algorithms，1987,82-89

[28] Nirmalakhandan,N.Gomputer-aided tutorials and tests for use in distance learning.Water Science and Technology,20D4,49(8):65-71

[29] David Lippiatt,Traditional,Distance and Virtual:an exploration of concepts.International Workshop on Distance Learning and Virtual Campus,2000

[30] Dejan Sunderic.SQL Server 2000 Stared Procedure&XML Programming,Second Edition.U.S.A.:McGraw-Hi 1 I Osborne Medi a,2003

[31] 梁娜，禹农，杨国青，基于 B/S 计算模型的 Web 技术在电子商务中的应用.山东利技大学学报(自然科学版)2003， 01：29

[32] 王云丽，马永祥，曹金刚，基于 j2ee 的多层结构应用系统的开发，河北省科学院学报，2003，01：56

[33] 穆福森，吴观茂基于 Struts+Spring+Hibernate.Web 应用开发框架技术[J].电脑知识与技术 2006，（02）

[34] 鲁明，赖芳东，李明，C/S 与 B/S 模式在 j2ee 框架下的综合应用，微型电脑应用，2003，04：36

[35] 蔡明，陈永运，J2EE 架构的研究与应用，计算机应用与软件，2003.8：87

[36] 于晓慧，王移芝；J2EE 架构下分层结构的研究和中间件的设计［J］.铁路计算机应用，2003.7：13-14

[37] 陈丽芳，毛力.；J2EE 模式在企业级应用程序中的集成应用［J］. 微计算机信息，2006，8-3：143-144

[38] 崔成磊； 基于 Web、J2EE 技术的工作流系统 [D];电子科技大学; 2005 年

[39] 伊静，智英杰，刘培玉，孙玮；基于 J2EE 技术及 Struts 架构的电力营销系统的设计与实现 [J];信息技术与信息化; 2007 年 04 期

[40] BruceEckel 侯捷译.Java 编程思想(第 2 版)[M].北京：机械工业出版社，2002..

[41] 金舒元，戴亚非，杜跃进；基于 WEB 的数据库发布技术 [J];小型微型计算机系统; 1998 年 07 期

[42] 颜惠；ASP 技术和 SQL 语言及其在数据库动态查询中的应用 [J];情报杂志; 2002 年 11 期

[43] 林建，董亚波；基于 J2EE 的数据访问技术设计模式的研究[J].计算机工程与应用

[44] Scott Mitchell. Working with Data in ASP.NET 2.0:: Nested Data Web Controls. www. asp.

Net

[45] Rick,Stefan.. 系统的软件测试(Systematic Software Testing)[M]. 2002