安徽农业大学

硕士学位论文

基于CMMB网络的紧急广播信息发布系统的设计和实现

姓名：罗淦坤

申请学位级别：硕士

专业：农业机械化工程

指导教师：江家伍

20100601

Y1735103

摘要

我国是世界上受自然灾害危害最严重的国家之一。洪涝、干旱、台风、地震、沙尘暴、冰雹、龙卷风、低温冷害、山体滑坡、泥石流、森林和草原火灾等世界上常见的自然灾害在我国都有不同程度的发生。面对灾害的发生，人类目前还没有有效的科技手段完全加以阻止或消灭，但可以采取有效的防范和得当的处置措施，把灾害造成的损失降低到最小的程度。中国每年应各种自然灾害造成的经济损失约占国内生产总值(GDP)的3%～6%。科学有效的防灾减灾措施取决于及时准确地掌握和利用各类预警信息。如何将预警信息及时准确地传达到灾害发生地，最大程度地降低灾害损失，成为目前迫切需要解决的重大课题。

紧急广播业务是CMMB（China Mobile Multimedia Broadcasting）中国移动数字多媒体广播系统提供的一项扩展业务。紧急广播是一种利用广播通信系统迅速通告民众的业务。当发生突发事件危害公众生命财产安全时，例如遭遇战争、恶劣天气、洪水、传染病等灾害时，或当政府机构向公众通告政令时，紧急广播提供了一种迅速快捷的通讯方式。当发生自然灾害、事故灾难、公共卫生和社会安全等突发事件时，造成或者可能造成重大人员伤亡、财产损失、生态环境破坏和严重社会危害，危及公共安全时，移动多媒体广播的紧急广播系统可以提供一种迅速快捷的通告方式。CMMB紧急广播系统采用卫星广播，不受高山、森林等地理障碍的影响，大大缩短了灾害预警信息的传播时间，为突发紧急公共事件赢得宝贵的应急响应时间，是建设和谐社会的具体体现。

国家广播电影电视总局为提高广播预警能力，以上海为试点，开展了CMMB紧急广播信息发布系统建设工作，并成立了项目组进行开发。本论文以该项目为背景。试图建立起基于CMMB网络的信息发布系统，在出现国家规定的紧急情况和突发事件时，可以实现向用户提供紧急通告服务，紧急广播以文字信息为基本方式，可扩展支持音频、图片、图像等方式，应实现多语种和多编码，以适应少数民族地区和在境内的境外人员，当前端发送紧急广播消息时，接收终端应能强行切换接收紧急广播消息，并能根据紧急事件的级别和类型采取相应的展示方式以达到预警目的。

关键字：CMMB, 紧急广播信息，信息发布，审核

Abstract

Our country is harmed by the natural disaster one of most serious national in the world. Floods, droughts, typhoons, earthquakes, dust storms, hail, tornadoes, cold damage, landslides, mudslides, forest and grassland fires that the common natural disaster has the varying degree occurrence in our country. Facing disaster's occurrence, the humanity did not have the effective technical method to perform at present to prevent or the elimination completely, but may adopt the effective guard and the appropriate handling measure, causes the losses the disaster to reduce to the smallest degree. Those natural disasters cause the economic loss approximately composes GDP (GDP) 3%~6% every year in our country. How to forewarn the information to transmit accurately promptly to the disaster birthplace, the greatest degree reduces the disaster to lose, becomes the important issue which present urgent needs to solve.

Emergency broadcasting is the CMMB (China Mobile Multimedia Broadcasting) mobile digital multimedia broadcasting system to provide a business expansion in China. Emergency Broadcasting is known as a way for Mobile Multimedia System to broadcast emergent events to public. When emergency such as natural disasters, traffic accident, events about public health and safety occurs, resulting in heavy casualties or property losses, or even damage to environment and social, in a word, has become great threat to public safety, Emergency Broadcasting System of Mobile Multimedia System can provide a convenient way of informing people in danger. CMMB Emergency Broadcasting System transmit signals by means of satellites broadcasting, whose transmission couldn't be affected by geography barrier such as mountain or forest, shortens transmission time of disaster warning signals a lot, and can win valuable response time of these urgent public events. can be recognized as a concrete manifestation of construction of harmonious society.

The state administration of radio film and television for the enhancement broadcast early warning ability, take Shanghai as an experiment site, has done the CMMB emergency broadcast information issue system construction work, combines to make or become initiated a project an item of group to carry on the development. The paper take this project as a background. Attempts to establish based on the CMMB network information issue system, when presents the national stipulation the

emergency case and the thunderbolt, may realize to the user provides the urgent circular service, the urgent broadcast take the writing information as the basic mode, may expand ways and so on support audio frequency, picture, image, should realize multilingual and the multi-codes, adapts the national minority area and in the domestic foreign personnel, when front end transmits the urgent broadcast message, receives the terminal to be able cut the receive urgent broadcast message forcefully, and can act according to emergencies' rank and the type adopts the corresponding demonstration way to serve the early warning purpose.

Key words: CMMB, emergency broadcasting message, information issued, audit

独 创 性 声 明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，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，也不包含为获得安徽农业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了谢意。

研究生签名：罗金艳　　时间：2010年 06月 17日

关于论文使用授权的说明

本人完全了解安徽农业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件和磁盘，允许论文被查阅和借阅，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。同意安徽农业大学可以用不同方式在不同媒体上发表、传播学位论文的全部或部分内容。

(保密的学位论文在解密后应遵守此协议)

研究生签名：罗金艳　　时间：2010 年06月17日

第一导师签名：[illegible]　　时间：2010 年 6月17日

插图清单

图 1 1 移动媒体广播电视系统总体构成图.....10
图 2 1 MVC 模式架构图.....16
图 2 2Struts2 架构图.....17
图 2 3Spring 架构图.....19
图 2 4 AOP 的各个基本概念以及相互关系[22].....25
图 2 5 Hibernate 架构图.....26
图 3 1 系统高层用例图.....34
图 3 2 系统部属情况.....36
图 4 1 系统框架架构图.....40
图 4 2 系统架构流程图.....41
图 4 3 引入分层模式后的系统架构图.....44
图 4 4 引入了切面管理事务的系统架构图.....45
图 4 5 紧急广播消息生命周期.....46
图 4 6 紧急广播消息审核流程.....47
图 4 7 广播消息管理用例图.....48
图 4 8 广播消息管理主界面.....49
图 4 9 广播信息添加页面.....49
图 4 10 广播信息审核页面.....50
图 4 11 消息区域管理用例图.....51
图 4 12 广播区域管理主界面.....51
图 4 13 网关管理用例图.....52
图 4 14 网关管理主界面.....52
图 4 15 系统状态监控管理主界面.....53
图 4 16 日志管理主界面.....54
图 4 17 数据管理主界面.....54
图 4 18 系统权限模块模型图.....55
图 4 19 账号管理模块用例图.....55
图 4 20 账号管理主界面.....56
图 4 21 系统 ER 图.....59
图 5 1 Message 与 Area 的关联对象模型.....61
图 5 2 MessageDAO 类图.....67

图 5　3 系统各层次之间的调用图.....68

附表清单

表 2-1 PersonDAO.java 以及 PersonServiceImpl.java 类 13
表 2-2 PersonDAOImpl.java 和 PersonServiceImpl.java 在 spring 文件中的配置 13
表 2-3 spring 配置文件中 PersonService 类的调用 14
表 2-4 setter 方式依赖注入 14
表 2-5 构造方式依赖注入 14
表 2-6 User.java 类的 Hibernate hbm 配置文件 18
表 2-7 Organization 类与 Person 类一对多关联关系映射 19
表 2-8 Person 类与 Organization 类多对一关联映射 20
表 3-1 系统运行硬件环境 27
表 3-2 系统运行软件环境 27
表 5-1 Message.hbm.xml 配置文件 54
表 5-2 Area.hbm.xml 配置文件 56
表 5-3 MsgArea.hbm.xml 配置文件 58
表 5-4 MessageService.java 部分代码 61
表 5-5 MessageService 类与 MessageDAO 类解耦的 Spring 配置文件 62
表 5-6 事务在 Spring 文件中的配置 63
表 5-7 MessageAction 的 struts 配置文件 64

1.绪论

1.1. 选题背景以及研究意义

1.1.1. CMMB 概述

CMMB 是英文 China Mobile Multimedia Broadcasting 的缩略语简称，意为中国移动多媒体广播电视。CMMB 是英文 China Mobile Multimedia Broadcasting 的缩略语简称，意为中国移动多媒体广播电视。移动多媒体广播业务是指通过卫星和地面无线广播方式,在移动便携手持式终端上,实现随时随地接收音视频广播电视节目和信息服务等业务。国际上比较成熟的移动多媒体广播标准包括:欧洲的 DVB2H 标准、韩国的 T2DMB 标准、美国的 Media2FLO 标准、日本的 ISDB2T 标准以及日本和韩国的 S2DMB 标准等。2006 年 10 月 24 日,国家广播电影电视总局正式颁布了 CMMB (中国移动多媒体广播)系统的广播信道行业标准 GY/T 220.1—2006《移动多媒体广播第一部分:广播信道帧结构、信道编码和调制》，传输技术采用我国自主研发的 STIM I(卫星地面交互式多业务体系)标准。

CMMB 采用“天地一体”的技术体系，即：利用大功率 S 波段卫星覆盖全国 100%国土、利用地面覆盖网络进行城市人口密集区域有效覆盖、利用双向回传通道实现交互，形成单向广播和双向互动相结合、中央和地方相结合的无缝覆盖的系统[1]。CMMB 的总体构成如图 1-1 所示。

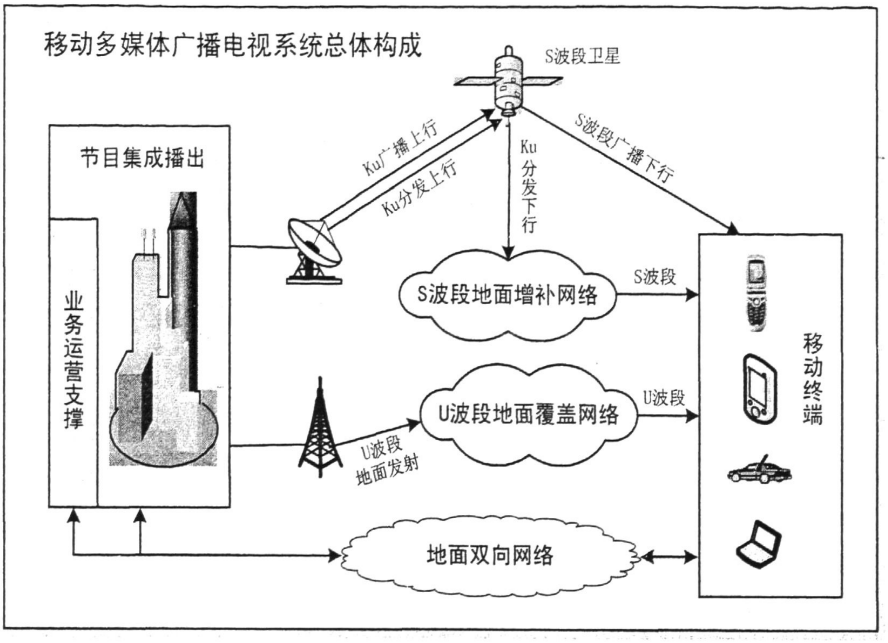

图 1　1 移动媒体广播电视系统总体构成图

diagram 1　1 The general structure diagram of the cmmb system

在 CMMB 的系统构成中，CMMB 信号主要由 S 波段卫星覆盖网络和 U 波段地面覆盖网络实现信号覆盖。S 波段卫星网络广播信道用于直接接收，Ku 波段上行，S 波段下行；分发信道用于地面增补转发接收，Ku 波段上行，Ku 波段下行，由地面增补网络转发器转为 S 波段发送到移动终端。为实现城市人口密集区域移动多媒体广播电视信号的有效覆盖，采用 U 波段地面无线发射构建城市 U 波段地面覆盖网络[2]。

移动多媒体广播电视系统主要面向手机、PDA 等小屏幕、移动便携手持式终端，车载电视等终端提供广播电视服务。根据移动多媒体广播电视业务发展需要，移动多媒体广播电视业务平台主要由公共服务平台、基本业务平台、扩展业务平台等三个平台构成。

（1）公共服务平台是向用户提供公益服务的移动多媒体广播电视业务平台，主要由公益类广播电视节目和政务信息、紧急广播信息构成。CMMB 公共服务平台播出的内容和开展的业务，为向合法用户提供的无偿服务。

（2）基本业务平台是向用户提供基本数字音视频广播服务和数据服务的业务平台，包括卫星平台和地方平台传送的数字音视频广播服务和数据服务。CMMB 基本业务平台向合法用户提供的服务，为有偿服务。

（3）扩展业务平台是根据用户不同消费需求向用户提供扩展广播电视节目服务和综合信息服务的业务平台。CMMB 扩展业务平台提供的服务主要由四方面构成，一是经营类的广播电视付费节目；二是经营类的音视频点播推送服务，利

用系统闲置时间将用户定制的广播电视节目推送到用户终端；三是综合数据信息服务，主要有股票信息、交通导航、天气预报、医疗信息等；四是双向交互业务，主要有音视频点播、移动娱乐、商务服务等。目前，移动多媒体广播电视主要以音视频服务为主，扩展服务中综合信息、双向交互等服务将随着移动多媒体广播电视业务的发展逐渐推广应用。CMMB 扩展业务平台向合法用户提供的服务，为有偿服务。

1.1.2. 自然灾害以及突发事件的破坏性

我国是世界上受自然灾害危害最严重的国家之一。洪涝、干旱、台风、地震、沙尘暴、冰雹、龙卷风、低温冷害、山体滑坡、泥石流、森林和草原火灾等世界上常见的自然灾害在我国都有不同程度的发生。在各类自然灾害中，气象灾害占70%以上。据统计，中国每年因各种气象灾害造成的经济损失约占国内生产总值(GDP)的3%一6%[3]。

2003 年，SARS 疾病的影响，仅中国大陆被确证病例 5327，死亡病例 348，死亡率 6.5%，而且对经济影响非常巨大。2005 年 11 月 13 日 13 时 40 分左右，地处吉林省吉林市的中石油吉林石化公司 101 厂一化工车间连续发生爆炸。5 人死亡，1 人失踪，70 人受伤。2005 年 11 月 26 日，江西九江发生 5.7 级地震，余震波及数省 50 多个县市，截至 27 日已造成 13 人死亡，8000 多人受伤；2006 年 6 月 17 到 2006 年 6 月 20，福建再遭暴雨到特大暴雨袭击，发生严重灾情，直接经济损失约 7 亿，四十四点六二万人受灾，倒塌房屋近一万五千间，因灾死亡二人、失踪十七人；农作物受灾二十二点四四千公顷，成灾面积十四点五八千公顷，绝收近三千公顷。

以上列举的实例表明，气象灾害，突发事件的发生对国家和个人造成的损失时不可估量的。我国地域辽阔、地形复杂、人口散居，特别是偏远山区，利用地面无线电视广播或有线电视广播是难以覆盖的[4]。CMMB 紧急广播系统采用卫星广播，不受高山、森林等地理障碍的影响，大大缩短了灾害预警信息的传播时间，为突发紧急公共事件赢得宝贵的应急响应时间，是建设和谐社会的具体体现。面对灾害的发生，人类目前还没有有效的科技手段完全加以阻止或消灭，但可以采取有效的防范和得当的处置措施，科学有效的防灾减灾措施取决于及时准确地掌握和利用各类预警信息，把灾害造成的损失降低到最小的程度已成为目前迫切需要解决的重大课题。

1.1.3. 选题背景以及研究意义

国家广播电影电视总局为提高广播预警能力，以上海为试点，开展了 CMMB

紧急广播信息发布系统建设工作，并成立了项目组进行开发。本论文以该项目为背景。试图建立起基于CMMB网络的信息发布系统，在出现国家规定的紧急情况和突发事件时，可以实现向用户提供紧急通告服务，紧急广播以文字信息为基本方式，可扩展支持音频、图片、图像等方式，应实现多语种和多编码，以适应少数民族地区和在境内的境外人员，当前端发送紧急广播消息时，接收终端应能强行切换接收紧急广播消息，并能根据紧急事件的级别和类型采取相应的展示方式以达到预警目的。

为此我们依托现有CMMB网络，一方面拓展紧急通知信息发布渠道，增加信息发布内容，建设针对不同群体的发布接收子系统，终端包括手机、电视、PAD、车载移动电视、移动便携终端等。解决预警信息发布“最后一公里”瓶颈问题。同时另一方面整合现有的各种发布通道和系统，建立统一的紧急广播信息发布流程，最终建设一个权威、畅通、有效的紧急广播信息发布系统。

通过一个权威、客观、畅通、可信的渠道使社会公众及时、准确、客观、全面地了解气象预警信息，有利于消除公众疑虑或事先掌握灾害预测信息，避免不必要的损失或把灾害造成的损失降低到最小的程度[5]，因此，建设CMMB紧急广播信息发布系统不仅具有深刻的理论价值而且具有巨大的实际意义。

1.2. 国内外研究状况

1.2.1. 国外研究状况

目前发达国家在灾害预警等紧急广播信息发布方面有很多成功经验，值得我们学习和借鉴。

美国国家海洋和大气管理局（NOAA）隶属于美国国家商务部，其气象电台（NWR）是全国范围内的无线电广播网络，以“美国国家海洋和大气管理局全灾害气象电台（NOAA All Hazards Weather Radio）”而著称。它直接从国家气象局（NWS）办公室获取天气信息，以无线电广播的形式持续地发布信息。气象电台一天 24 小时播报国家天气服务预报所提供的预警、警戒、预报和其他灾害信息。在紧急状态下，气象电台会打断正常的广播，通过特殊的声音启动当地天气播报[6]。

气象电台包括940多个无线电发射机，涵盖了美国的50个州（全部）、近海水域、波多黎各、美属维尔京群岛和美属太平洋海域。气象电台发布的灾害广播，用普通的调幅调频收音机是收不到信号的，需要特殊的无线电接收机或超声扫描仪来接收信号。在七个频率的特高频（VHF）公共服务波段可以接收广播信息。

气象电台与美国联邦通信委员会（FCC）主办的紧急预警系统的合作，使其

成为一个可以广播“所有灾害”的无线电广播网络，可以只通过这一个来源了解综合的天气和紧急情况信息。通过与联邦政府、各州、各地方的紧急事件管理者以及其他的公共服务官员的联合，气象电台还可以播报所有类型的意外事件的预警信息和事后处理信息，包括自然灾害（比如地震、雪崩）、环境灾害（比如化学物质泄漏、石油喷发、核电站的紧急状态）、公众安全信息（比如美国失踪人口紧急广播警报、911 电话停用信息）、国家紧急状态（比如：恐怖袭击）信息等。

美国天气频道 TWC (The Weather Channel）始建于 1982 年，是美国惟一提供 24 小时不间断天气服务的国家有线电视网，有地区性、全国性以及全世界的天气预报。除午夜的 2 个小时之外，美国天气频道全天处于直播状态，任何相关警报可以无延时地进行发布，对影响大的灾害进行现场直播，让社会了解灾害的进程以及应急措施。每隔十分钟，发布一次全国范围的温度、降水、雷暴等预报信息，可能引起灾害的加以警示。频道节目内容丰富，有天气新闻、气象灾害分析、气象科普、气象课堂、气象和经济，还有重大灾害性天气预报和警报等[7]。

目前该频道的用户数已超过 9000 万户，覆盖 90%以上的有线电视用户。在全美 45 个有线频道中，收视满意度名列榜首，入户率第一，综合排名第二。1999 年 9 月，飓风 Floyd 登陆美国，美国天气频道现场追踪报道飓风的收视率超过 CNN 对海湾战争报道的收视率。

美国联邦通信委员会的紧急报警系统（Emergency Alert System 简称 EAS）建立于 1994 年 11 月。该系统与数千个广播电（视）台、有线电视系统以及卫星公司相连，可以在全国紧急状况下向公众传送信息。美国州以上的广播电（视）台和有线电视系统每周都要对紧急报警系统进行调用。州及州以下的每月调用一次，并结合当地的实际情况进行信息发布[8]。

从 1997 年 1 月 1 日开始，美国所有的调频调幅广播电台和电视台都使用上述调用程序。1998 年 12 月 31 日开始， 1 万用户以上的有线电视系统也加入了紧急报警系统，能够在所有的视频频道中发送紧急信息。

此外，美国许多州目前使用现存的空袭警报系统（Air Raid sirens）来发布龙卷风和大洪水的警报。居住在某些核设施附近的人们在家中使用特制的收音机，这些收音机用来在辐射泄漏等紧急情况下广播紧急信号。有些紧急情况也可通过电子邮件和手机短信发布应急信息。

1.2.2. 国内研究状况

我国在预警信息发布方面已经有不少成功的技术和经验，建立了相对比较完善的气象灾害预警等紧急广播信息发布系统，手机短信、12121 固定电话、有线

广播前端接收机（农村大喇叭）、海洋短波广播电台、电子显示屏、气象频道等是目前各级政府和广大人民群众获取预警信息的主要手段。

虽然具备上述多种手段进行预警信息的发布，但存在以下问题：

(1)各种气象预警等紧急广播信息发布系统及相关发布通道各自独立；工作人员要多次登录不同的发布系统进行操作；预警信息重复录入和管理；地理信息和接收人群信息在多个发布系统中互相不能共享导致这些公共信息不能进行统一有效的存储、管理、维护和访问；

(2)我国地域辽阔、地形复杂、人口散居，特别是偏远山区，而现有的系统的均是采用互联网或者手机发射塔来进行接发信息的，利用互联网或者手机发射塔是难以覆盖到偏远山区，这样就不能保证将最新的紧急广播信息发布出去。紧急广播信息发布存在"最后一公里"和"最后一小时"的瓶颈[9]。

1.3. 本论文主要研究的内容概述

紧急广播是一种移动多媒体系统向公众通告紧急事件的方式。当发生自然灾害、事故灾难、公共卫生和社会安全等突发事件时，造成或者可能造成重大人员伤亡、财产损失、生态环境破坏和严重社会危害，危及公共安全时，移动多媒体广播的紧急广播系统可以提供一种迅速快捷的通告方式[10]。CMMB 紧急广播信息发布系统采用卫星广播，不受高山、森林等地理障碍的影响，大大缩短了灾害预警信息的传播时间，为突发紧急公共事件赢得宝贵的应急响应时间，是建设和谐社会的具体体现。

文章的第一部分主要是简要介绍本论文的选题背景、国内外发展状况，以及论文的组织与内容介绍。

文章的第二部分简要介绍系统中所采用的技术，其中介绍 Struts2 框架、Spring 框架以及 Hibernate 框架。

第三部分主要论述 CMMB 紧急广播信息发布系统的需求分析，通过对国家广播电影电视总局提出的需求进行分析，确定了系统所需要实现的基本功能，确定系统中主要的数据流程。

第四部分主要论述 CMMB 紧急广播信息发布系统架构的建立过程，对系统中的类按照职能进行分类，根据类职能的不同对系统进行分层并介绍架构模式在 CMMB 紧急广播信息发布系统架构设计中的应用。同时对实现系统主要功能所需要的功能模块进行设计。最后对 CMMB 紧急广播信息发布系统中的数据库进行设计。

第五部分主要论述 CMMB 紧急广播信息发布系统的实现。其中按照系统中的各个层次来分别介绍。

第六部分对本文的工作进行了总结，提出一些存在的问题以及进一步工作的展望。

2.轻量级框架分析概述

2.1. 呈现层 struts 框架简述

2.1.1. MVC 模式

模型－视图－控制器（Modal View Controler）是 Xerox PARC 在八十年代为编程语言 Smalltalk－80 发明的一种软件设计模式[11]，至今已被广泛使用。Modal 是指数据模型，View 是指用户界面，Controler 则是控制器。使用 MVC 的目的是将 Modal 和 View 的实现代码机型分离，使每个部分的功能更加单一更加明确，层次更分明，为后期维护提供了方便。MVC 的体系机构如图 2-1 所示[12]：

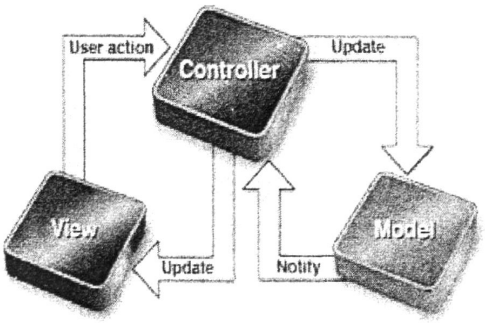

图 2 1 MVC 模式架构图

diagram 2 1 MVC model structure diagram

模型层用于封装与应用程序的业务逻辑相关的数据以及对数据的处理方法。“模型”有对数据直接访问的权力，例如对数据库的访问。“模型”不依赖“视图”和“控制器”，也就是说，模型不关心它会被如何显示或是如何被操作。但是模型中数据的变化一般会通过一种刷新机制被公布。为了实现这种机制，那些用于监视此模型的视图必须事先在此模型上注册，从而，视图可以了解在数据模型上发生的改变。

视图层能够实现数据有目的的显示（理论上，这不是必需的）。在视图中一般没有程序上的逻辑。为了实现视图上的刷新功能，视图需要访问它监视的数据模型（Model），因此应该事先在被它监视的数据那里注册。

控制器起到不同层面间的组织作用，用于控制应用程序的流程。它处理事件并作出响应。“事件”包括用户的行为和数据模型上的改变[13]。

MVC 处理过程：首先控制器接收用户的请求，并决定应该调用哪个模型来

进行处理；然后模型根据用户请求进行相应的业务逻辑处理，并返回数据；最后控制器调用相应的视图来格式化模型返回的数据，并通过视图呈现给用户。

2.1.2. MVC 模式的 Struts2 实现

Struts2 是在 Webwork 的基础上发展而来，在 Struts2 中，Model 对应业务逻辑组件，它通常用于实现业务逻辑方法以及以及与底层数据库的交互等；View 对应视图组件，通常是指 JSP 页面，但也适用于其他视图显示技术，如 Velocity 或者 Excel 文档；Control 对应系统核心控制器和业务逻辑控制器，系统核心控制器为 Struts2 框架提供的 FilterDispatcher，它根据请求自动调用相应的 Action。而业务逻辑控制器是指开发人员自行定义的一系列 Action，在 Action 中负责调用相应的业务逻辑组件来完成处理。Struts2 系统架构如图 2-2 所示：

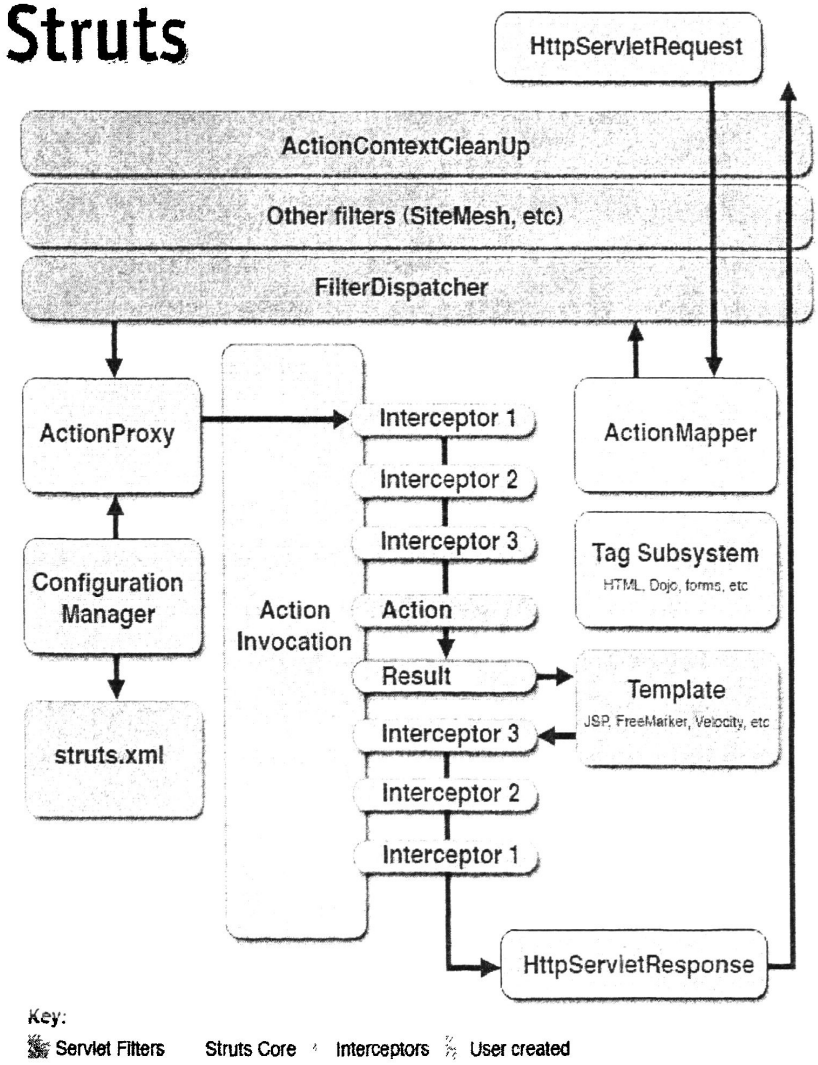

图 2　2Struts2 架构图

diagram 2-2 Struts2 structure diagram

一个请求在 Struts2 框架中的处理大概分为以下几个步骤[14]：

(1)客户端初始化一个指向 Servlet 容器（例如 Tomcat）的请求；

(2)这个请求经过一系列的过滤器（Filter）（这些过滤器中有一个叫做 ActionContextCleanUp 的可选过滤器，这个过滤器对于 Struts2 和其他框架的集成很有帮助，例如：SiteMesh Plugin）；

(3)接着 FilterDispatcher 被调用，FilterDispatcher 询问 ActionMapper 来决定这个请是否需要调用某个 Action；

(4)如果 ActionMapper 决定需要调用某个 Action，FilterDispatcher 把请求的处理交给 ActionProxy；

(5)ActionProxy 通过 Configuration Manager 询问框架的配置文件，找到需要调用的 Action 类；

(6)ActionProxy 创建一个 ActionInvocation 的实例；

(7) ActionInvocation 实例使用命名模式来调用，在调用 Action 的过程前后，涉及到相关拦截器（Intercepter）的调用；

(8)一旦 Action 执行完毕，ActionInvocation 负责根据 struts.xml 中的配置找到对应的返回结果。返回结果通常是（但不总是，也可 能是另外的一个 Action 链）一个需要被表示的 JSP 或者 FreeMarker 的模版。在表示的过程中可以使用 Struts2 框架中继承的标签。在这个过程中需要涉及到 ActionMapper。在上述过程中所有的对象（Action，Results，Interceptors，等）都是通过 ObjectFactory 来创建的。

Struts2 的目标很简单就是使 Web 开发变得更加容易。为了达成这一目标，Struts2 中提供了很多新特性，比如智能的默认设置、annotation 的使用以及“惯例重于配置”原则的应用，而这一切都大大减少了 XML 配置。Struts2 中的 Action 都是 POJO，这一方面增强了 Action 本身的可测试性，另一方面也减小了框架内部的耦合度，而 HTML 表单中的输入项都被转换成了恰当的类型以供 action 使用。开发人员还可以通过拦截器（可以自定义拦截器或者使用 Struts2 提供的拦截器）来对请求进行预处理和后处理，这样一来，处理请求就变得更加模块化，从而进一步减小耦合度。模块化是一个通用的主题。可以通过插件机制来对框架进行扩展；开发人员可以使用自定义的实现来替换掉框架的关键类，从而获得框架本身所不具备的功能；可以用标签来渲染多种主题（包括自定义的主题）；Action 执行完毕以后，可以有多种结果类型包括渲染 JSP 页面，Velocity 和 Freemarker 模板等[14]。

2.1.3. Struts2 的优点[15]

(1)简化 Action，Struts2 中 Action 摇身变为普通的 JAVA 类，任何具有 execute()

方法的类都可以作为 Struts2 的 Action，不必实现某个接口、继承某个类；

(2)取消了 ActionForm 类，Action 可以直接获得用户参数；

(3)改进返回值，不同于 ActionForward，Struts2 的 Action 能返回多种类型的数据；

(4)Struts2 的标签库提供了对 JSF、Freemarker 的支持；

2.2. 业务逻辑层 Spring 框架概述

2.2.1. Spring 概述

Spring 是基于 MVC 模型的作用于业务层的开源框架。简单的说，Spring 就是实现了 AOP 功能的 IoC 容器，虽然说法不全面但是概括了 Spring 架构的核心功能[16]。相对于 EJB，Spring 框架提供一个轻量级 J2EE 框架实现，具有分层清楚、速度快、代码少，支持最新的 ORM 技术与 AOP 概念等优点。Spring IoC、Spring AOP、Spring 服务抽象三者一起形成了 Spring 这样机体整体，并且使得构建轻量级的 J2EE 框架体系变的更容易。Spring 框架由 7 个定义良好的模块组成[17]。

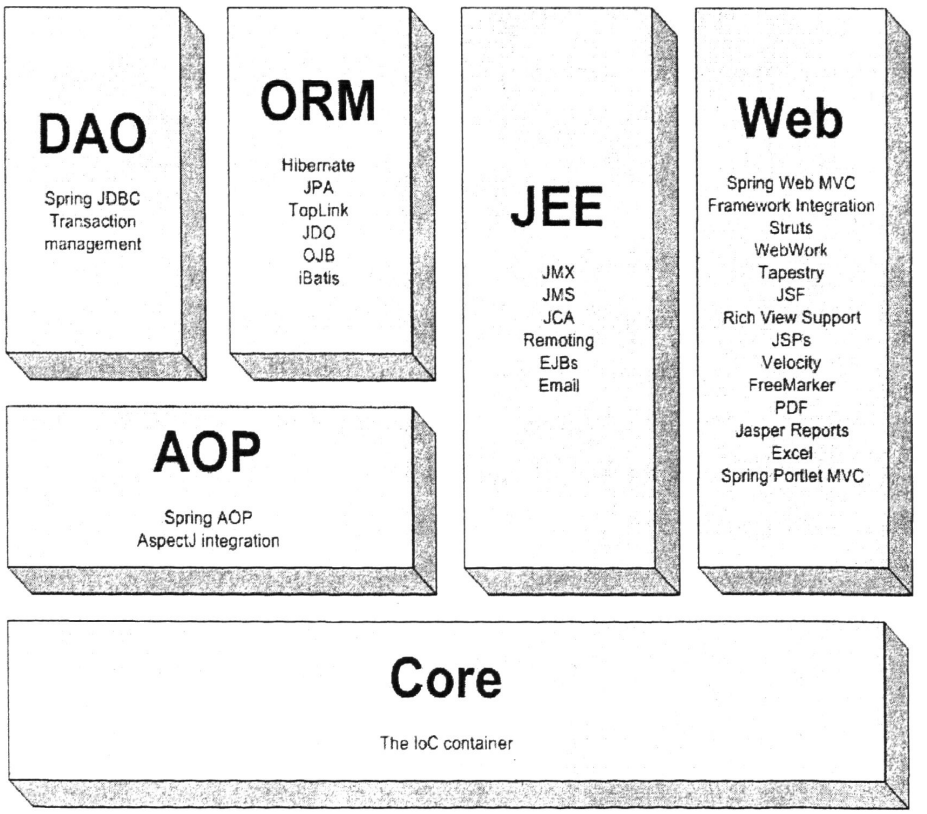

图 2　3Spring 架构图

diagram 2-3 Spring structure diagram

每个模块的功能如下[18]：

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

Spring Context：Spring 上下文是一个配置文件，向 Spring 框架提供上下文信息，比如企业服务，例如 JNDI、电子邮件、国际化、调度功能等。

Spring AOP： Spring AOP 模块直接将面向方面的编程功能集成到了 Spring 框架中。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP，不用依赖 EJB 组件，就可以将声明性事务管理集成到应用程序中。

Spring DAO：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。 并且，JDBC 封装包还提供了一种比编程性更好的声明性事务管理方法，不仅仅是实现了特定接口，而且对所有的 POJOs（plain old Java objects）都适用。

Spring ORM：封装提供了常用的“对象/关系”映射 API 的集成层。 其中包括 JPA、JDO、Hibernate 和 iBatis 。利用 ORM 封装包，可以混合使用所有 Spring 提供的特性进行“对象/关系”映射，如前边提到的简单声明性事务管理。

Spring Web 模块：Web 上下文模块建立在应用程序上下文模块之上，为基于 Web 的应用程序提供了上下文。所以，当与 Web Work 或 Struts 一起使用 Spring 时，这个包使 Spring 可与其他框架结合。Web 模块还简化了处理多部分请求以及将请求参数绑定到域对象的工作。

Spring Web MVC：MVC 框架是一个完整的搭建 Web 应用程序的 MVC 实现。通过配置接口，MVC 框架变成为高度可配置的，MVC 容纳了大量视图技术，其中包括 JSP、Velocity、Tiles、iText 和 PoI 等。

总之，Spring 框架可以用在现在大部分 J2EE 服务器中，大多数功能也能适用于不同的开发环境。Spring 的核心要点是：支持不绑定到特定 J2EE 服务的可重用业务和数据访问对象。这样的框架组件可以在不同 J2EE 环境(Web 或 EJB)、独立应用程序、测试环境之间重用，这也是轻量级框架的最大优势之一。

2.2.2. IoC 介绍

Spring 能有效地组织 J2EE 应用各层的对象。不管是控制层的 Action 对象，还是业务层的 Service 对象，还是持久层的 DAO 对象，都可在 Spring 的管理下有机地协调运行。这都可以都归功于 Spring 的核心机制，依赖注入。

依赖注入(Dependency Injection)和控制反转(Inversion of Control)是同一个概

念。IoC 的原理是基于面向对象设计原则的好莱坞理论：别找我们，我们会找你哦[19]。具体含义是：当某个角色(比如一个 Java 实例)需要另一个角色(另一个 Java 实例即被调用者)的协助时，创建被调用者的工作不再由调用者来完成，创建被调用者实例的工作通常由 Spring 容器来完成，然后注入调用者；这个工作原理称为控制反转，也称为依赖注入。

IoC 在 Spring 通过以下步骤进行配置和调用，首先在 Spring 的配置文件中配置对象以及依赖的对象；然后通过架构中的 BeanFactory 类，来自动实现上文所述注入过程，以 PersonServiceImpl.java 与 PersonDAO.java 为例，如下表所示：

表 2　1 PersonDAO. java 以及 PersonServiceImpl. java 类

table 2-1 PersonDAO. java class and PersonServiceImpl. java class

```
public class PersonServiceImpl {
    private PersonDAO personDAO;
    public void setPersonDAO(PersonDAO personDAO){
        this.personDAO = personDAO;
    }
    public void save(Person person){
        this.personDAO.save(person);
    }
}
public class PersonDAO {
    public void setPersonDAO(PersonDAO personDAO){
        this.personDAO = personDAO;
    }
    public void save(Person person){
        this.personDAO.save(person);
    }
}
```

Spring 的配置文件为如下表所示：

表 2　2 PersonDAOImpl. java 和 PersonServiceImpl. java 在 spring 文件中的配置

table2-2 configuration of PersonDAOImpl. java and PersonServiceImpl. java in the spring

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
```

```
  <bean id="personService" class="com.hynoo.service.impl.PersonServiceImpl"
        scope="singleton">
    <property name="personDAO" ref="personDAO"/>
  </bean>
  <bean id="personDAO" class="com.hynoo.dao.impl.PersonDAOImpl"
        scope="singleton"/>
</beans>
```

最后，调用的过程如下表所示：

表 2　3 spring 配置文件中 PersonService 类的调用

table2-3 the class of PersonService class from spring configuration file

```
BeanFactory factory=new ClassPathXmlApplicationContext(new String[]
{"beans.xml"});
PersonService personService=(PersonService)factory.getBean("personService");
personService.save(new Person());
```

在 IoC 中，一个对象需要依赖其他的对象，依赖关系由 IoC 容器注入到组件中去，常见的 IoC 注入形式有以下两种：构造器依赖注入和 setter 依赖注入[20]。

setter 依赖注入：使用 setter 方法注入的时候，主控对象通过属性来表达自己所依赖的对象和所需配置的值。Java 实现中，只要对象提供 JavaBean 标准的属性就可以了。例如：PersonServiceImpl 类依赖 PersonDAO 对象的注入，为它定义一个 setter 方法，该方法接受类型为 PersonDAO 的参数,程序代码如下表所示：

表 2　4 setter 方式依赖注入

table2-4 IoC of the setter

```
public class PersonServiceImpl {
    private PersonDAO personDAO;
    public void setPersonDAO(PersonDAO personDAO){
        this.personDAO = personDAO;
    }
}
```

构造器依赖注入：通过具体类的构造方式判断如何将 PersonDAO 实例注入到 PersonServiceImpl 类。因此，PersonServiceImpl 类必须声明一个构造函数，并在其中包含所有需要注入的元素。程序代码如下表所示：

表 2　5 构造方式依赖注入

table2-5 IoC of construtor

```
public class PersonServiceImpl{
    private PersonDAO personDAO;
    public PersonServiceImpl(PersonDAO personDAO){
```

```
        this.personDAO = personDAO;
    }
}
```

2.2.3. AOP 介绍

面向切面编程（AOP）提供另外一种角度来思考程序结构，通过这种方式弥补了面向对象编程（OOP）的不足。 除了类（classes）以外，AOP 提供了切面。切面对关注点进行模块化，例如横切多个类型和对象的事务管理。

AOP 的基本思想是把横跨多个模块的行为从各个模块中分离出来，并封装为一个新的模块，可以单独对它编程、修改，在需要时使用编织器把这个模块编织到原有的代码中，能克服 OOP 不能处理跨越多个不相关的行为的缺陷[21]。AOP 的核心就是将应用程序中的业务逻辑处理部分同对其提供支持的通用服务，即对“横切关注点”进行分离，这些“横切关注点”贯穿了程序中的多个纵向模块的需求。一个完整的 AOP 定义包含有下面的一些基本概念[18]：

切面（Aspect）： 一个关注点的模块化，这个关注点可能会横切多个对象。事务管理是 J2EE 应用中一个关于横切关注点的很好的例子。在 Spring AOP 中，切面可以使用通用类（基于模式的风格） 或者在普通类中以 @Aspect 注解（@AspectJ 风格）来实现。

连接点（Joinpoint）： 在程序执行过程中某个特定的点，比如某方法调用的时候或者处理异常的时候。 在 Spring AOP 中，一个连接点 总是 代表一个方法的执行。通过声明一个 org.aspectj.lang.JoinPoint 类型的参数可以使通知（Advice）的主体部分获得连接点信息。

通知（Advice）： 在切面的某个特定的连接点（Joinpoint）上执行的动作。通知有各种类型，其中包括“around”、“before”和“after”等通知。 通知的类型将在后面部分进行讨论。许多 AOP 框架，包括 Spring，都是以拦截器做通知模型， 并维护一个以连接点为中心的拦截器链。

切入点（Pointcut）： 匹配连接点（Joinpoint）的断言。通知和一个切入点表达式关联，并在满足这个切入点的连接点上运行（例如，当执行某个特定名称的方法时）。 切入点表达式如何和连接点匹配是 AOP 的核心：Spring 缺省使用 AspectJ 切入点语法。

引入（Introduction）： （也被称为内部类型声明（inter-type declaration））。声明额外的方法或者某个类型的字段。 Spring 允许引入新的接口（以及一个对应的实现）到任何被代理的对象。 例如，你可以使用一个引入来使 bean 实现 IsModified 接口，以便简化缓存机制。

目标对象（Target Object）： 被一个或者多个切面（aspect）所通知（advise）的对象。也有人把它叫做 被通知（advised） 对象。 既然 Spring AOP 是通过运行时代理实现的，这个对象永远是一个 被代理（proxied） 对象。

AOP 代理（AOP Proxy）： AOP 框架创建的对象，用来实现切面契约（aspect contract）（包括通知方法执行等功能）。 在 Spring 中，AOP 代理可以是 JDK 动态代理或者 CGLIB 代理。注意：Spring 2.0 最新引入的基于模式（schema-based）风格和@AspectJ 注解风格的切面声明，对于使用这些风格的用户来说，代理的创建是透明的。

织入（Weaving）： 把切面（aspect）连接到其它的应用程序类型或者对象上，并创建一个被通知（advised）的对象。 这些可以在编译时（例如使用 AspectJ 编译器），类加载时和运行时完成。 Spring 和其他纯 Java AOP 框架一样，在运行时完成织入。

通知的类型：

前置通知（Before advice）： 在某连接点（join point）之前执行的通知，但这个通知不能阻止连接点前的执行（除非它抛出一个异常）。

返回后通知（After returning advice）： 在某连接点（join point）正常完成后执行的通知：例如，一个方法没有抛出任何异常，正常返回。

抛出异常后通知（After throwing advice）： 在方法抛出异常退出时执行的通知。

后通知（After (finally) advice）： 当某连接点退出的时候执行的通知（不论是正常返回还是异常退出）。

环绕通知（Around Advice）： 包围一个连接点（join point）的通知，如方法调用。这是最强大的一种通知类型。 环绕通知可以在方法调用前后完成自定义的行为。它也会选择是否继续执行连接点或直接返回它们自己的返回值或抛出异常来结束执行。

AOP 的各个概念以及相互关系如图 2-4 所示：

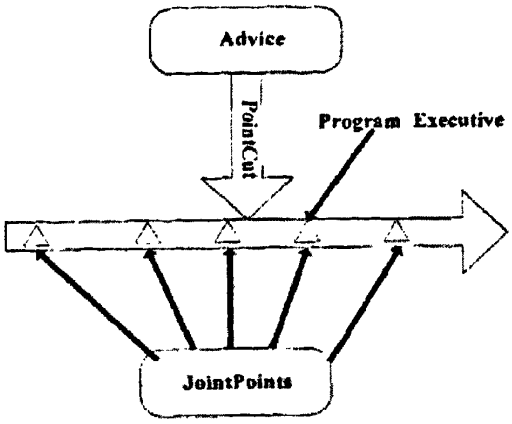

图 2 4 AOP 的各个基本概念以及相互关系[22]

diagram2-4 the basic concepts of AOP, and the relationship

Spring 中 AOP 的实现就是通过动态代理来实现的。权限验证以及事务管理机制是 Spring 中目前最为实用的 AOP 应用。

2.3. 数据持久层 Hibernate 框架概述

2.3.1. ORM 概念

ORM，即 Object-Relational Mapping，就是 Java 应用中对象到关系数据库中的表的自动的转化，使用元数据对对象与数据库之间的映射进行描述[23]。这样，在具体的操作数据库的时候，就不需要再去和复杂的 SQL 语句打交道，只要像操作普通对象一样操作它就可以了，进而解决了面向对象开发与面向关系型数据的阻抗不匹配的问题。

2.3.2. ORM 的优势

(1)提高开发效率。减轻了开发人员编写数据库访问代码的工作量，使得他们可以将更多的注意力放在商业逻辑上[24]。因为在实际的开发中，真正对客户有价值的是系统中关键高效的业务逻辑服务。

(2)提高代码性能。Hibernate 本质上是包装了 JDBC 来进行数据库操作的，由于 Hibernate 在调用 JBDC 上面是优化 JDBC 调用，并且尽可能的使用最优化的，最高效的 JDBC 调用，所以性能令人满意。

(3)Hibernate 降低可系统对数据库产品的依赖，有效提高系统的可靠性。

2.3.3. Hibernate 体系结构

Hibernate 技术是一个提供数据库服务的中间件。它的架构如图 2-5 所示[25]：

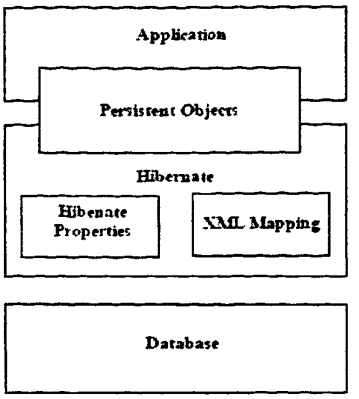

图 2 5 Hibernate 架构图

diagram 2-5 Hibernate structure diagram

从这个图可以看出，Hibernate 使用数据库和配置信息来为应用程序提供持久化服务（以及持久的对象）。

2.3.4. Hibernate 映射机制[26]

Hibernate 常用映射有四种：基础数据映射、继承关系映射以及关联关系映射、组件映射，在 CMMB 紧急广播信息方法系统中只用到了基础数据映射和关联关系映射，以下分别简略介绍：

基础数据映射：基础数据是关于基本类型属性的数据，基础数据在软件系统中广泛使用[27]。ORM 工具实现应用程序中从类对应到表、属性对应到列、关系对应到外键、Java 类型对应到数据库类型的映射。

Hibernate 中使用 XML 的数据的映射。例如把代码中创建的一个 User 类映射到关系数据库中的 cmmb_user 表中，关键 XML 配置代码如下表所示：

表 2 6 User.java 类的 Hibernate hbm 配置文件

table2-6 the configuration of User.java in the hibernate hbm configuration file

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.hynoo.cmmb.user.pojo.User" table="cmmb_user"
catalog="cmmb">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="native" />
```

```
        </id>
    <property name="username" type="java.lang.String"/>
     </class>
</hibernate-mapping>
```

其中<?xml version="1.0" encoding="utf-8"?>为 Hibernate 映射 DTD 必须在每个映射文件中声明， 它是 XML 的校验句法要求的[28]。<hibernate-mapping>表示是映射元素的声明，它可以声明多个要映射的类，包含在<class> 元素中。<class name>这行表示类 com.hynoo.cmmb.user.pojo.User 被映射为数据库中的表 cmmb_user，表中的每一行代表类的一个实例。<id>元素中的属性是对象的标志属性，即这个对象区别于其他对象的属性， 标志属性映射为数据库表中的主键；property 则是一般属性， 比如<property name="username" type="java.lang.String"/>映射了表 cmmb_user 中一个数据表的 username 字段，它的类型为：String。

以上是一个简单的基本映射文件，对象/关系数据库的映射便是以这个 hbm.xml 文件为基准， 便可以编写应用程序对持久层进行访问。

关联映射：管理类和类之间的关联关系是 ORM 的核心任务之一。最为常见映射是的一对多、多对一关联[29]。比如有两个类 Organization 和 Person，一个 Organization 对应多个 Person，因此 Organization 对 Person 是一对多关联，Organization 类中有一个属性为 person。在 Organization 的映射文件中用<one-to-many/>表示，它把 Organization 类对应表中的主键映射为 Person 类对应表中的外键，具体代码如下表所示：

表 2 7 Organization 类与 Person 类一对多关联关系映射

table2-7 the one2many relationship of between Organization and Person

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.hynoo.cmmb.pojo.Organization" table="cmmb_organization"
catalog="cmmb">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="native" />
        </id>
    <set name="persons" inverse="true" lazy="true" cascade="all-delete-orphan">
```

```
        <key column="person"/>
        <one-to-many class="com.hynoo.cmmb.pojo.Person">  //一对多关联关系
    </set>
    </class>
</hibernate-mapping>
```

对于 Person 来说，与 Organization 是多对一的关联。为了保持关系的双向性，在 Person 的映射文件中也必须用一个<many-to-one>来表示，具体代码如下表所示：

表 2 8 Person 类与 Organization 类多对一关联映射

table2-8 the many2one relationship of between Person and Organization

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.hynoo.cmmb.pojo.Person" table="cmmb_person"
catalog="cmmb">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="native" />
        </id>
    <many-to-one name="organization" class="com.hynoo.cmmb.pojo.Organization"
column="person">    //多对一关联映射
    </class>
</hibernate-mapping>
```

Person 与 Organization 之间的关联是通过 Organization 的主键 person 完成的，同时它也是 Person 表的外键。为了把两个文件中定义的关联当作同一个关联关系来看，用一个 inverse 属性进行标志，inverse 表示控制反转，关联关系由多的一方来控制，在<one-to-many>关系的 set 元素中加入<set inverse="true">。cascade 的属性值为 all-delete-orphan 表示子实体被持久化了，则父实体也被持久化了；父实体被删除，子实体也被删除。

2.3.5. Hibernate 查询机制

传统的 SQL 语言采用的是结构化的查询方法，它是基于面向关系类型的数

据库的，并不支持直接以对象形式的数据进行存储。Hibernate 提供了一种语法类似于 SQL 的语言，Hibernate 查询语言(HQL)。和 SQL 不同的是，HQL 是一种面向对象的查询语言，它可以查询以对象形式存在的数据。HQL 是 Hibernate 官方推荐的 Hibernate 检索方式，它使用类似 SQL 的查询语言，以面向对象的方式从数据库中查询。可以使用 HQL 查询具有继承、多态和关联关系的数据[30]。

虽然 HQL 的语法类似于 SQL，但实际上它的查询目标是对象。HQL 是非常有意识的被设计为完全面向对象的查询，它可以理解如继承、多态 和关联之类的概念。HQL 拥有面向对象语言的所有的特性，这其中包括多态、继承和组合。这就相当于一个面向对象的 SQL，为了提供更强大的功能，HQL 还提供了很多的查询函数：投影函数、约束函数、聚合函数、分组函数等，这和 SQL 语法很相似。所以对 SQL 语句熟悉的话很容易就能对 HQL 语言进行编程。

此外 HQL 还有一些要和 SQL 有区别的技术细节[31]：

(1)HQL 并不区分字母的大小写，但在 HQL 中的 Java 类和属性名必须和实际的类和属性名一致。如 SELECT 和 select 之间可以互换，但 Order 和 order 却代表不同的含义。

(2)如果 HQL 中引用的类未被导入，在 HQL 中必须引用具体的包等。

最后要考虑的是 Hibernate 的性能优化的问题。在数据库操作中，Hibernate 框架需要将执行的语句转换为对应的数据库 SQL 语句，然后才执行。所以 Hibernate 的效率一般都低于直接 JDBC 存取。有些情况下就是对实现同一功能的语句，用 Hibernate 与用 JDBC 性能相差几倍以上，所以性能优化调整，也是数据库开发模块的关键。对于 Hibernate 性能上的调优，主要优化点有:数据库设计调整、HQL 优化、Hibernate API 的正确使用、还可以对主配置参数优化，比如：日志，查询缓存，fetch_size,batch_size 等，还可以在映射文件上进行优化如：ID 生成策略，二级缓存，延迟加载，关联优化等。其中二级缓存是最有效的方法，针对二级缓存还有许多其他策略比如：事务控制策略、数据库设计、降低关联的复杂性、尽量不使用联合主键 ID 的生成机制等等。但是由于不同的数据库所提供的机制不完全一样，所以 Hibernate 性能的优化方法不是固定不变的，不同的数据库环境和具体开发背景对优化方法有不同的要求，这需要开发人员的有灵活的应对策略。

3. 需求分析

3.1. 功能性需求

本节将对广电总局对 CMMB 紧急广播信息发布系统提出的系统建设目标，得出系统需要完成的任务即系统的功能性需求，并确定系统的参与者，最后给出系统的高层用例图。

3.1.1. 系统建设目标分析

国家广播电影电视总局“CMMB 紧急广播信息发布系统”项目的建设目标如下：

(1)ESG 信息编辑(ESG: electronic service guide 电子业务指南)

ESG 信息包括 ESG 基本描述信息、ESG 数据信息和节目提示信息。ESG 信息编辑功能进行 ESG 基本描述信息、ESG 数据信息的编辑，节目提示信息由 ESG 系统自动生成。

(2)ESG 信息编辑主要包括：

1)对 ESG 基本描述表的网络级别、网络号、本地时间偏移、字符编码类型、ESG 业务标识 ServiceID、数据类型、数据块及数据块版本等信息进行编辑和维护；

2)对 ESG 数据信息的业务信息、业务扩展信息、编排信息、内容信息和业务参数信息进行编辑和维护；

3)支持多用户同时在线编辑，用签入签出机制保障操作安全性；

4)ESG 数据信息应支持动态传送，对编辑后的数据信息能自动进行重新封装和传送。

(3)ESG 信息审核

节目审核人员对编辑好的 ESG 信息进行预览、审核，确认无误后，发布 ESG 信息。只有审核通过的 ESG 信息才能进行后续的封装和发送。审核功能包括：

1)提供审核界面，对已编辑提交的 ESG 数据进行审核，决定该 ESG 数据是否发送；

2)将审核结果进行数据存储；

3)未经过审核或审核未通过的 ESG 数据不能发送；

4)审核通过的 ESG 数据不能进行再次编辑；

5)提供 ESG 数据的预览功能。

(4)参数配置

参数配置包括目标复用器地址和端口参数配置、基本描述表参数配置和 XML 根元素参数配置。

移动多媒体广播 ESG 发生器应具有目标复用器地址和端口参数配置功能，参数配置包括基本描述信息的目标复用器组播地址和端口；节目提示信息的目标复用器单播地址；数据信息的目标复用器单播地址。

移动多媒体广播 ESG 发生器应具有基本描述表参数配置功能包括对物理网络信息的管理，包括网络级别、网络号和本地时间偏移；对基本描述信息的配置，包括字符编码类型和 ESG 业务标识。

移动多媒体广播 ESG 发生器应具有对 XML 根元素中数据信息发布者和数据信息版权所有者进行参数配置的功能。

(5)权限管理

要求 ESG 系统提供操作员权限管理功能，能够对 ESG 编辑人员、审核人员和系统管理人员的使用权限进行约束配置。

(6)ESG 封装

ESG 系统按照 GY/T 220.3-2006 标准对 ESG 信息进行封装，封装主要内容为：

1)ESG 基本描述表封装

根据 ESG 更新序号、分配给 ESG 传输的业务标识、数据类型标识、数据块标识等参数，按照 GY/T 220.3-2006 标准“5.1 基本描述表”格式，构造、封装成 ESG 基本描述表。

2)节目提示信息封装

根据当前系统时钟，从节目编排信息中提取每个频道的节目提示信息，按照 GY/T 220.3-2006 标准“5.2 节目提示信息”语法进行封装。

3)ESG 数据信息封装

根据数据类型表示、压缩编码方式、分节信息等，对 ESG 各元素按照 GY/T 220.3-2006 标准“5.3 数据信息”语法进行封装。

(7)ESG 传送

ESG 系统按照 GY/T 220.3-2006 标准对 ESG 信息进行封装，并按照《移动多媒体广播复用实施指南》中 ESG 接口要求将消息发送给复用器。

(8)数据导入

支持运营商所规定的 XML、txt 或其他有明确定义的包含所要求信息的 txt 等格式的文件导入为中央节目 ESG 信息。

(9)数据导出

将中央节目 ESG 信息按照 GY/T 220.3-2006 标准“6 XML 数据模型”定义的格式，生成本地 xml 格式文件。

(10)播出控制

移动多媒体广播 ESG 根据设定的播控参数配置对 ESG 播出进行控制。包括：

1)基本描述表、节目提示信息按设置的发送间隔进行发送；

2)数据信息按设置的发送配置（ESG 数据发送天数、数据信息编码方式和数据信息发送选择）进行发送；

3)数据发送码率控制在设定的输出码率范围内。

相应的播控参数配置包括：基本描述表发送间隔；节目提示信息发送间隔；ESG 数据发送天数，缺省为 7 天；数据信息编码方式，缺省为 GZIP 压缩编码方式；数据信息发送选择：业务信息、编排信息和业务参数信息必须发送；业务扩展信息和内容信息可选择是否发送；输出码率。

(11)状态指示

可指示移动多媒体广播 ESG 的工作状态，发生异常情况时应能通过图、文、声音等方式及时报警。

(12)日志管理

日志管理包括日志产生和日志维护。

日志产生是对 ESG 发生的重要事件进行记录、保存，便于系统维护，记录的事件信息有：

1)系统操作事件，如 ESG 编辑、用户资料修改、ESG 审核发布等；

2)用户登录事件，如用户登录时间、登出时间等；

3)告警事件，如网络通信失效告警、网络同步时钟丢失告警、磁盘空间告警等。

系统日志维护包括对日志信息的查看、备份和删除。

(13)数据存储

数据存储功能包括：

1)存储物理网络信息和 ESG 基本描述信息；

2)存储原始 ESG 数据和经过审核的 ESG 数据；

3)存储用户资料信息和用户权限；

4)存储系统参数配置信息；

5)存储日志信息；

6)具有可控的手动或自动数据清理功能。

(14)时钟(可升级)

通过 NTP 协议，与网络时钟服务器建立通信连接，获取时钟信息，并同步调整本机的时钟，使移动多媒体广播 ESG 的各模块能参照网络同步时钟协同工作。

(15)与 C_SG 数据同步

实现与 C_SG 数据同步，要求符合《移动多媒体广播/手机电视业务总体技术方案》中的相关规定和接口定义。

根据对以上所叙述的客户目标的分析，可以确定短信服务平台将具有以下功能：广播消息管理、广播区域管理、网关管理、系统状态监控、日志管理、告警管理、数据维护、账号管理、时钟同步管理。

3.1.2. 系统参与者识别

参与者通常是代表一群通过与系统交互来获取价值的人，参与者也可以代表一个外部系统，它同坐与本系统的交互来获取价值或者为系统提供价值[32]。依据 CMMB 紧急广播系统中用户所具有的职责的不同，将系统用户分为以下三类

(1)系统管理员

系统管理员可以对系统中除信息发布功能以外的所有功能的操作。主要负责管理系统基本信息。

(2)信息发布员

信息发布员可以对紧急通知的信息进行发布、修改等与紧急广播信息相关的功能进行管理。

(3)信息审核员

信息审核员可以对发布员发布的信息进行审核，审核通过的紧急广播信息才能通过系统发布到终端。

3.1.3. 系统高层用例图

用例图从外部用户的角度捕获系统、子系统或类的行为，它将系统功能划分为对活动者具有意义的事务[33]。通过上一节的分析可知，在 CMMB 紧急广播信息发布系统中需要有对整个系统的基本信息进行维护的系统管理员，需要有对紧急广播信息维护的信息发布员，需要有对紧急广播信息发送进行审核的审核员。CMMB 紧急广播信息发布平台的高层用例图如图 3-1 所示：

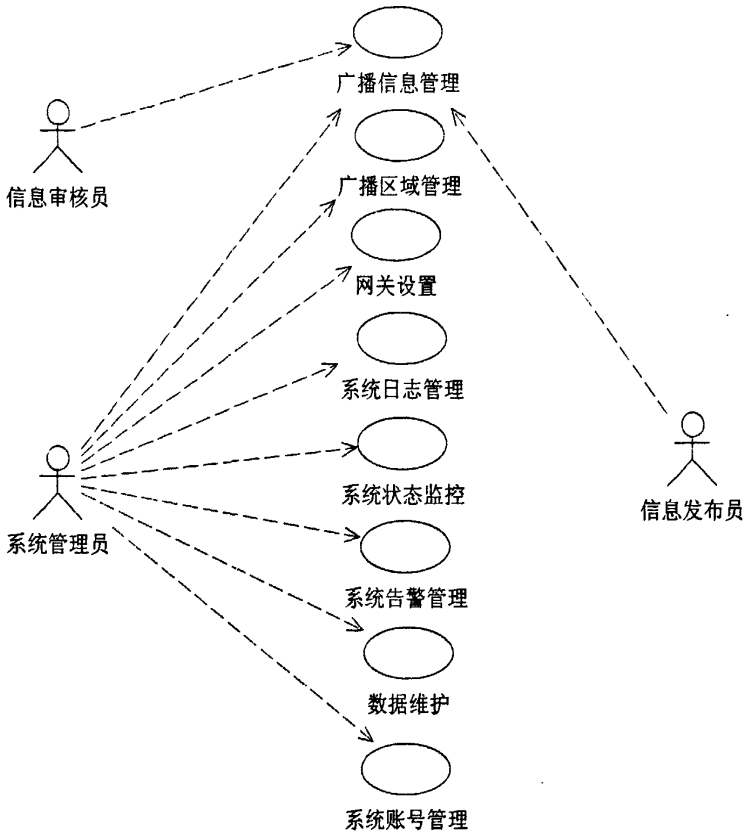

图 3 1 系统高层用例图

diagram3-1 use case diagram of system high level

3.2. 非功能性需求

3.2.1. 系统性能要求

CMMB 紧急广播信息发布系统除了要满足业务管理和发展各方面的实际需要，还必须能够稳定运行。只有这样才能确保应用部门和技术维护部门的工作效率，这是系统能否成功应用的重要保证。稳定性具体体现为：系统主机能够 7*24 小时不间断的持续工作；在系统出现故障能够有一定的处理措施保证关键业务能够继续运行。系统应具有应急备份机制，在主系统发生故障时自动启用备份系统进行工作。

3.2.2. 安全性要求

(1)系统应具备访问权限的识别和控制功能。对系统管理人员、ESG 编排人员、

ESG 审核人员及其他管理员授予不同的管理权限。要保证只有授权的人员或系统可以访问某种功能，获取和修改业务数据；

(2)系统应提供操作日志记录功能和操作日志查看功能；

(3)系统应提供完整的 ESG 审核功能，只有通过审核的 ESG 信息才能被播出到移动多媒体广播网络中；

(4)系统应具有多层次的防病毒能力。防病毒软件应具备全面查杀病毒，查杀病毒准确无误，管理方便，病毒特征码自动更新，安装简单的特点。

3.3. 系统运行环境

3.3.1. 硬件环境

表 3 1 系统运行硬件环境

table3-1 system running the hardware environment

要求项	最低配置	推荐配置
CPU	1 GHz Pentium IV	2×2.7 GHz Pentium IV
内存	512 MB	2 GB
硬盘	硬盘剩余容量 2 GB	硬盘剩余容量 10 GB
光驱	CDROM	CDROM
网卡	10 M/100 M 网卡	100 M 网卡
声卡	无特殊要求	无特殊要求
显卡	支持分辨率 1024×768	支持分辨率 1024×768

3.3.2. 软件环境

表 3 2 系统运行软件环境

table3-2 system operating software environment

要求项	推荐配置
操作系统	Windows Server2003 或更高版本
数据库	MySQL5.0

3.3.3. 系统的实际部属图

系统在网络中的实际部属情况如图 3-2 所示：

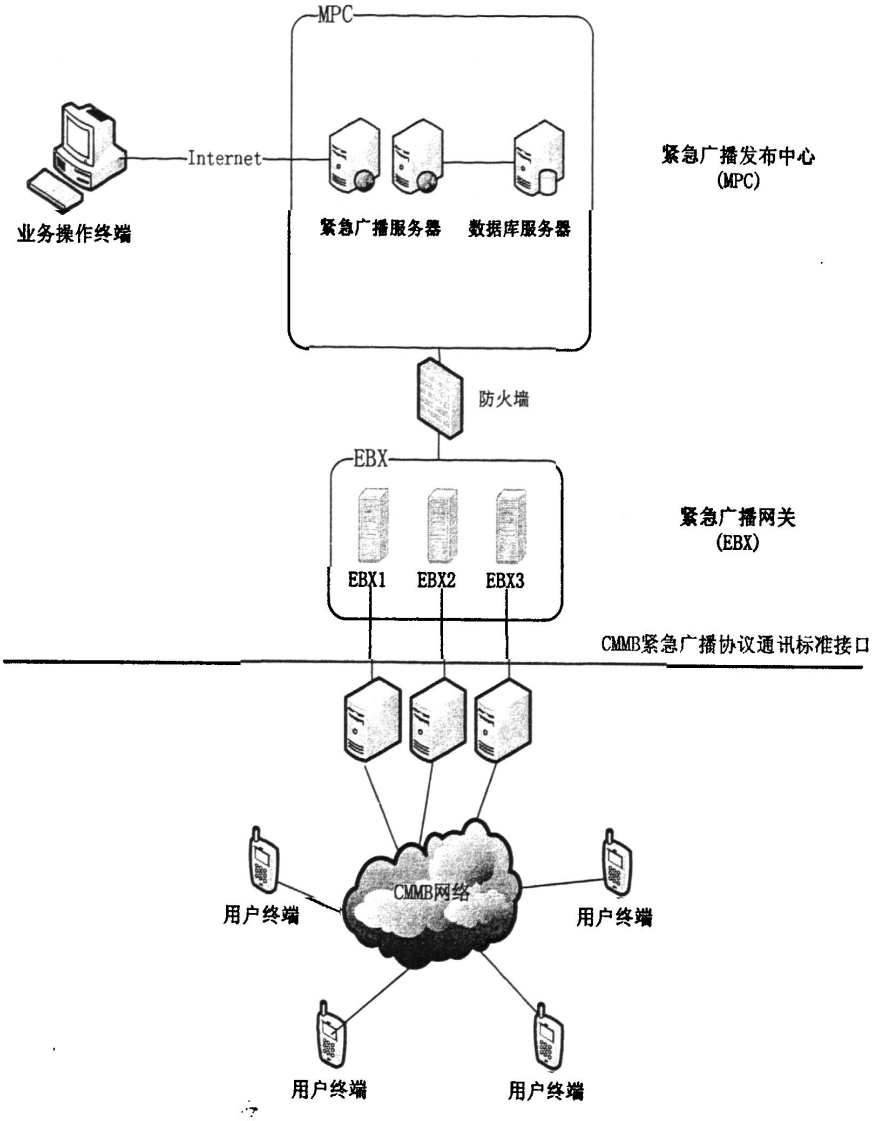

图 3　2 系统部属情况

diagram 3-2 the deploy of the system

3.4. 系统设计目标

3.4.1. 广播消息审核规范，认证严谨

所有紧急广播消息须由消息发布员提交，然后由消息审核员进行审核，审核通过后才准予发送至卫星复用器，由卫星复用器传送至卫星进行广播。

3.4.2. 广播条件灵活

可以设定紧急广播消息的广播范围；可以设定紧急广播消息的事件类型；可以预定紧急广播消息的广播时间；可以定制紧急广播消息的优先级、紧急广播的

持续时间、紧急广播的发送间隔。

3.4.3. 维护管理方便

安装时设定系统参数即可正常运行，安装后可以通过修改配置文件进行参数配置；可实时监测网关的运行状态；可自动记录系统的事件；可自动记录系统告警信息；可对不同状态的紧急广播消息进行分类管理和监测。

3.4.4. 软件扩展性强

在数据库的构建中，采用数据库中间件，来保证异构平台和各个应用子系统的数据升级和整合。遵循 J2EE 规范，采用面向对象设计和采用接口编程，确保功能扩充方便。

3.4.5. 可靠性高

系统监测完善，包括：网关的运行状态、信息发布中心事件、信息发布中心告警的管理；操作系统重启后，信息发布系统可自动运行，不需人工干预。与卫星复用器之间的连接发生故障后会自动重新连接，不需人工干预。

3.4.6. 安全性高

用户执行所有操作前必须经过权限验证；用户权限管理灵活，可自由设定用户的操作权限。

3.5. 系统设计原则

3.5.1. 可靠性和可用性

可用性要求所采用的软件技术必须是成熟的技术，相应的产品也必须是成熟的产品，确保业务应用系统能够稳定、安全、可靠地运行。障情况下，支持整体服务器迁移，通过自动和手动把集群化的服务器实例和以它为宿主的所有服务从一台计算机上迁移到另一台计算机上，确保集群化服务器的故障恢复。

3.5.2. 可扩展性

应用支撑平台的框架是可扩展的，能够适应业务和技术的发展趋势。通过保证当前应用的可重用性，最大程度地保护投资。

3.5.3. 开放性

本系统考虑到未来功能的扩充，产品方案能满足其技术开放性和二次开发的需求。基于业界开放式标准，遵循开放的技术标准（如 J2EE 标准、Web Services 标准、JSR-168 等），为未来的系统扩展奠定基础。

3.5.4. 技术先进性

系统设计充分考虑技术的先进性，采用先进成熟的技术，确保系统的技术先进性，保证用户投资的有效性和延续性。

3.5.5. 安全性

提供多种安全手段，如用户访问控制、信息加密等数据安全保护措施和系统安全保证。提供用户管理、身份认证、授权、等安全机制，确保只有授权用户才能访问系统资源。提供各个层次的安全访问和用户权限控制，确保整个系统的安全可靠运行。

4.系统设计

本章主要介绍CMMB紧急广播信息发布系统的设计。首先介绍了CMMB紧急广播信息发布系统架构的设计过程，并应用设计模式对系统架构进行改进。然后介绍了系统功能模块的设计以及数据库的设计。

4.1. 系统架构设计

一个软件系统的架构将系统中数量众多、种类各异的类结合起来，形成了一个有机的整体，并且展示了各部分之间结构上的相互关系，软件架构使得开发人员能够对系统有一个完整而连贯的认识。一个清晰、稳定的架构能够驱散疑惑，确立明确的开发目标。就像建筑蓝图能够有效地将木匠、电工、泥瓦匠分组，使他们能够有效地分工协作，共同完成一个更宏大的目标一样，软件架构也能够使各种开发人员在一个共同的目标下，基于对解决方案相同的理解的基础上彼此协调、合作[32]。

当开发一个软件架构时，需要遵循两条基本的原则：高内聚性和低耦合性[32]。

4.1.1. 系统架构的建立

系统架构的建立一般要经过确定系统技术架构、将类分组等一系列过程。

(1)确定系统技术架构

本系统采用模块化设计方式，采用J2EE轻量级架构。J2EE轻量级架构具体技术实现是利用Spring框架提供的IoC机制和AOP思想来整合Struts2和Hibernate架构。该J2EE轻量级架构在CMMB紧急广播信息发布系统中的使用，有助于实现系统的先进性、稳定性、扩展性、长期性。

采用Spring框架来组织紧急广播信息发布系统的业务逻辑，通过Spring的依赖注入模式，将依赖关系从编码中脱离出来，从而大大降低了各组件之间的耦合，实现了组件真正意义上的即插即用，将依赖关系从编码中脱离出来，从而大大降低了各组件之间的耦合。主要依赖于Spring的loC(反转控制，Inversion of Control)。它通过使用接收Spring管理的JavaBean所提供的Set方法(setter injection)或构造函数(constructor injection)把存在依赖关系的对象组织起来，为外部系统提供服务。这样，应用程序中的对象在不需要调用容器相关API的情况下就能获取它所依赖的对象的使用权。

系统采用 Struts2 作为表现层，有助于系统的扩展性，长期性。整个架构用 Struts2 作为 web 层的开发框架，统一管理视图和控制器部分，视图层用 JSP 表示，并在页面中严格限制 Java 代码和 Struts2 标签，利用 OGNL 实现视图层和模型层的分离，控制器用 Struts2 的命令对象 Action，并通过它最终完成控制器与业务逻辑层的交互。业务逻辑层使用自己编写的 Service 对象，利用接口与实现分离的方式在 Spring 架构中完成持久访问对象的注入和业务逻辑的事务管理，并关联系统的 web 层和持久层。

基于 Hibernate 框架的持久层，采用 DAO 设计模式完成最终的 Java 类和数据库之间的转换和访问。在实际开发中，持久化对象的映射文件还可以使用 Annotation 从业务对象类中直接生成。该框架的使用增强了系统的可移植性，Hibernate 可以屏蔽数据库的差异性。

采用上述方式，不仅利用 Spring、Hibernate 和 Struts2 框架各自的优势，而且成功地将视图、控制器与模型彻底分离，独立出来了业务逻辑层和持久访问层。另外，接口和实现分离的策略，也有助于把各个层之间的耦合度进一步降低，有利于系统的灵活性和可维护性，同时也便于多人同时并行开发，大大提高了项目的开发进度。系统框架架构如图 4-1 所示：

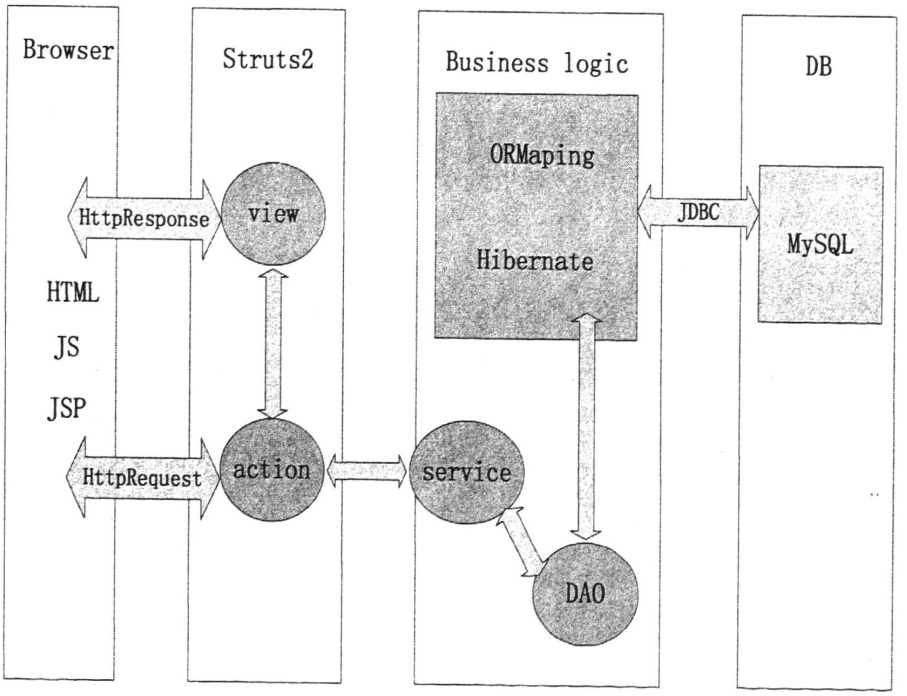

图 4　1 系统框架架构图

diagram4-1 System Framework Chart

1)系统架构的层结构：

表现层是一些带有 taglib 标签的 jsp 页面，主要是利用 JSP 技术 JavaScript

和 Html、Xml 结合 Struts2 强大的功能来实现。JSP 页面中不会嵌入 Java 代码，这样会大大增强代码的可读性和系统的可维护性。业务层是由 Struts2 的控制组件 Action 和业务逻辑类 JavaBean 来实现的。JavaBean 里封装了与数据持久层的交互关系，以实现所有的业务逻辑，该层运用了 Spring，通过依赖注入 IoC、面向方面编程 AOP、面向接口编程，来降低业务组件之间的耦合度、提高代码的重用性和增强系统扩展性。数据持久层开发中与数据库进行数据交互主要可以归为添加、读取、修改、删除，这些操作占据了系统开发中大部分的数据，同时还需要考虑与数据库交互的性能问题，如连接池、数据缓存等等。数据持久层主要利用 Spring 结合 Hibernate 来构建，其设计目标是为整个系统提供一个高层、统一、安全和并发的数据持久机制。

2)架构流程：

用户首先通过 Browser 发出 HttpRequest 请求，Http 请求会被架构业务层中的控制组件 FilterDispatcher 接受并将请求交给 Action 来处理，FilterDispatcher 根据请求的 URL 从 ActionMapper 得到相应的 Action。此外，Action 对 JSP 页面进行封装，如果需要的话还可以对表单中数据进行校验，如果出错就将返回到显示页面；校验通过，则将页面数据传递给 Action。调用 Action 的 excute 方法，该方法调用业务逻辑模块，由 Hibernate 和 Spring 在“幕后”共同完成数据库的交互。进行业务逻辑处理的 JavaBean 将通过数据持久层和域对象层与数据库进行交互。Action 的 excute 方法执行完毕后将返回字符串对象，并且会被 ActionSupport 接受，据此 Action 转发到字符串指定的视图。也就是将选择一个正确的视图显示给用户。整个处理过程如图 4-2 所示：

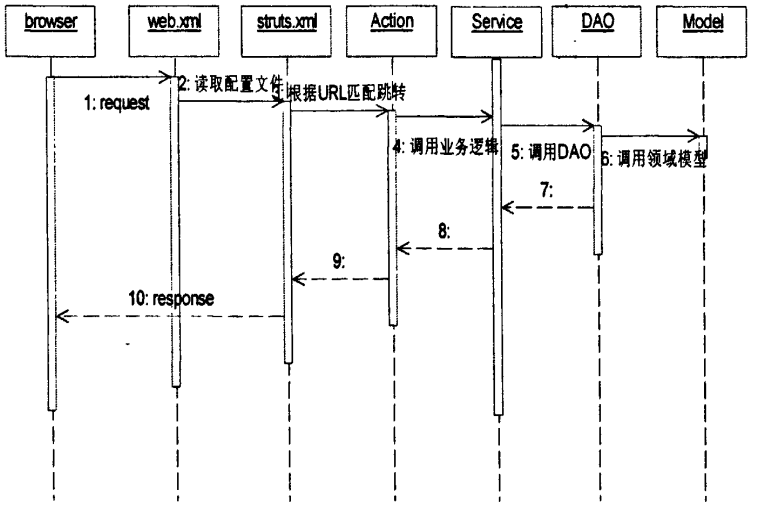

图 4　2 系统架构流程图

diagram4-2 System architecture diagram

3)架构优势：

轻量级容器提供了一种管理、定位业务对象的办，用不着 JNDI 寻址、定制服务器之类的额外辅助，轻量级容器为应用对象提供注册服务。其较之 EJB 容器而言，不仅功能强大，而且避免了容器强制业务对象采用特定的接口，最低程度的降低了侵入性，实现了效果极佳的架构重用。架构的整体设计采用了 MVC 设计模式，把表现层和业务层进行了很好的分离。轻量级容器中所有的 Java 类都运行在同一个虚拟机中。WEB 层是由 MVC 框架提供的 Struts2。业务对象是 POJO，运行在轻量级容器里。AOP 的拦截机制能够增强业务对象，从而实现企业级服务。数据访问机制通过轻量级的 ORMapping，该层能提供透明的持久化，该持久层实现了对数据访问方式 JDBC 的轻量级封装。并且由于 Spring 对 Hibernate 提供了很好的支持，二者结合可以构建优秀的数据持久层。该轻量级架构的使用不仅充分利用了各个框架自身的特点，而且最大限度地使用了复用技术，增强了系统的可维护性，明确了分工，降低了开发成本。

(2)将类分组[34]

可以为了使相互协作的类在一起，而将它们归到一个包内，也可以从职责相似的角度对类分组。在将类分组的过程中，必须在考虑可变性和重用的同时充分考虑耦合性和内聚性的因素。一般来说，那些能够在不同环境下进行重用的类应该按照职责将其组织到相同的包中。按照职责而不是协作关系进行分组有助于它们独立于特定的使用方式。专门针对一个协作的那些类必须与它们所支持类位于同一个包内。在 CMMB 紧急广播信息发布系统中将按照职能来对类进行分组。接下来将首先分析 CMMB 紧急广播信息发布系统中所需要具有哪些职能类来确立类的分组。

1)Model 类

在面向对象软件系统中，首先要用类来表示系统中的一些实体，这也称为实体类，实体类一般与现实世界中的具体事物实体或概念相对应，例如在 CMMB 紧急广播信息发布系统中的广播洗洗、发送区域、网关等都是实体类。实体类的主要职能为表示系统中具体的事物或概念。

1)Action 类

在 CMMB 紧急广播信息发布系统中涉及到大量操作数据的用户界面。这也是目前与 CMMB 紧急广播信息发布系统交互的主要接口。Action 类的主要职能是用来与外部系统进行交互。

2)Service 类

在 CMMB 紧急广播信息发布系统中，系统的所有业务逻辑都是通过 Service 来完成的。比如：在发送紧急通知的时候，既要发送一条紧急通知又要记录该紧急通知是由谁创建的，创建的时间等，这些都要保证在一个 Service 中完成，这

个 Service 交给 Spring 的事务进行管理，要么全部成功，要么就全部失败，只有交给 Spring 的事务才能保证数据库中不会有脏数据的存在。

3)DAO 类

DAO 类用来封装数据访问对象，只被 service 层调用，而不能被其他的类跨越调用。主要的职责是完成对 Model 类的增删改查。

4)数据源逻辑类

在 CMMB 紧急广播信息发布系统中必须对一些数据进行持久化，持久化不仅仅指保存数据的过程也包括对数据进行修改查询删除等操作。与数据库操作相关的类都可以归纳为数据源逻辑类。

5)辅助工具类

在 CMMB 紧急广播信息发布系统中，有些功能可以独立出来，在其它的系统中复用。例如文件的上传下载、日志管理功能以及文件导入导出，分页功能等。

4.1.2. 分层模式在系统架构中的应用

在上一节中按照所不同的职责将系统中的不同的类进行了分类。本节将利用分层模式将系统按照不同的职责进行分层，将具有同等职能的类划分到同一个层次中，增加层次内部的内聚性。在分解复杂的软件系统时经常采用的技术之一就是分层。当使用分层的观点来考虑系统时，可以将各个子系统想象成按照“多层蛋糕”的形式来组织，每一层都依托于其下层之上，在这种组织方式下，上层使用了下层定义的各种服务，而下层对上层一无所知。同时，每一层对自己的上层屏蔽其下层的细节[35]。

在 CMMB 紧急广播信息发布系统的架构设计中，也采用了分层的设计方案，将系统中的拨各个不同的模块按照分层的方式来组织。根据上文对 CMMB 紧急广播信息发布系统需求分析及将类按不同的职责分组的结果，在整个系统存在与用户或其他软件系统进行交互的用户接口层，因为在 CMMB 紧急广播信息发布系统中不涉及被其他系统调用，只是与用户进行交互，因此在这里称为表示层；需要处理紧急广播信息发送业务的业务逻辑层；需要处理数据访问对象的 DAO 层；需要将 CMMB 紧急广播信息发布系统中的数据持久化的数据持久层。业务逻辑层是整个系统的核心，表示层负责与系统用户交互，DAO 层完成对数据访问对象的封装，隔离 Service 和数据持久化层的直接调用，数据持久层完成数据的持久化操作。

按照不同的职责对系统进行分层，并面向接口编程降低层与层之间的依赖性，将系统中的可变性分别限制在四个层次中。由于表示层用户界面和业务逻辑层中的具体业务逻辑具有较大的可变性，当需要进行改变的时候只需要修改相应

的层次中的内容而不需要对其他层次进行改动，增加了层次内部的内聚性和降低系统的耦合性，上层对下层的访问将只通过接口，而不需要了解下层中的具体情况，进一步降低了上层对下层的依赖性，提高了系统的可维护性。引入了分层模式后的系统架构如图 4-3 所示：

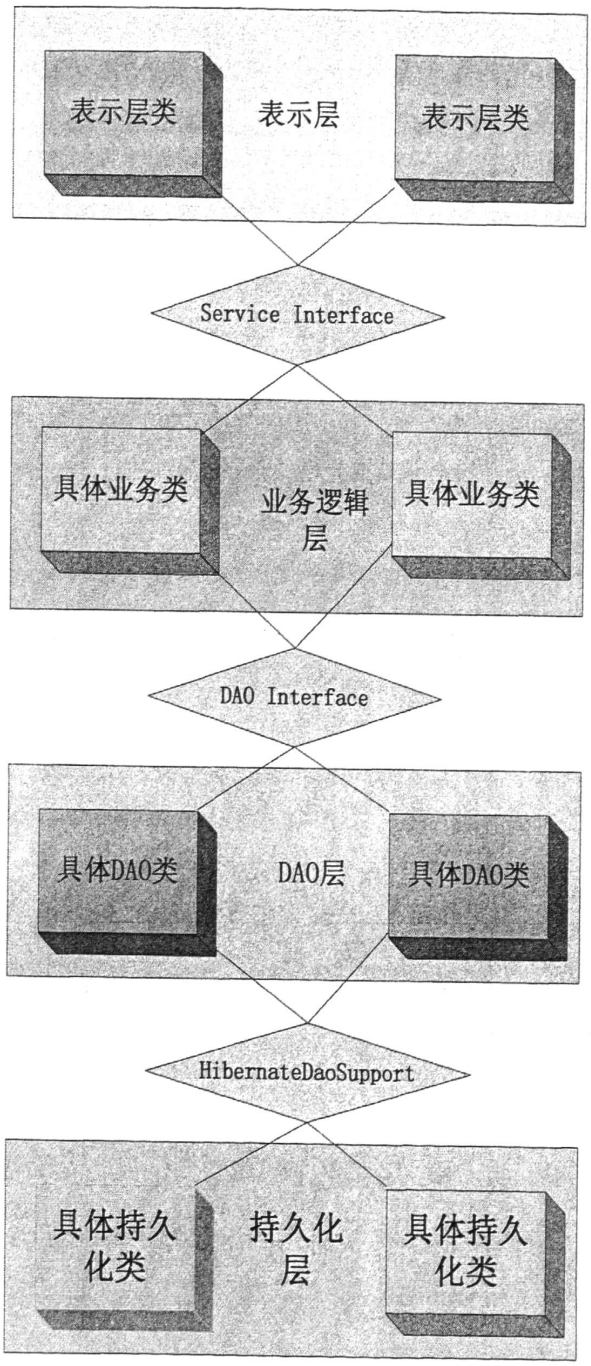

图 4 3 引入分层模式后的系统架构图

diagram4-3 the architecture diagram models after the introduction of layer

4.1.3. AOP 在系统架构中的应用

在本文的第二章中，曾经介绍了面向切面编程的概念。因为在 CMMB 紧急广播信息发布系统中涉及大量的事务管理过程。在系统中将事务管理作为切面从系统中的具体类中抽取出来，保证了系统中的类只处理与具体业务相关的操作，提高了组件的可重用性，并降低了交叉性业务和具体业务类之间的耦合。

引入了切面管理事务的系统架构如图 4-4 所示。

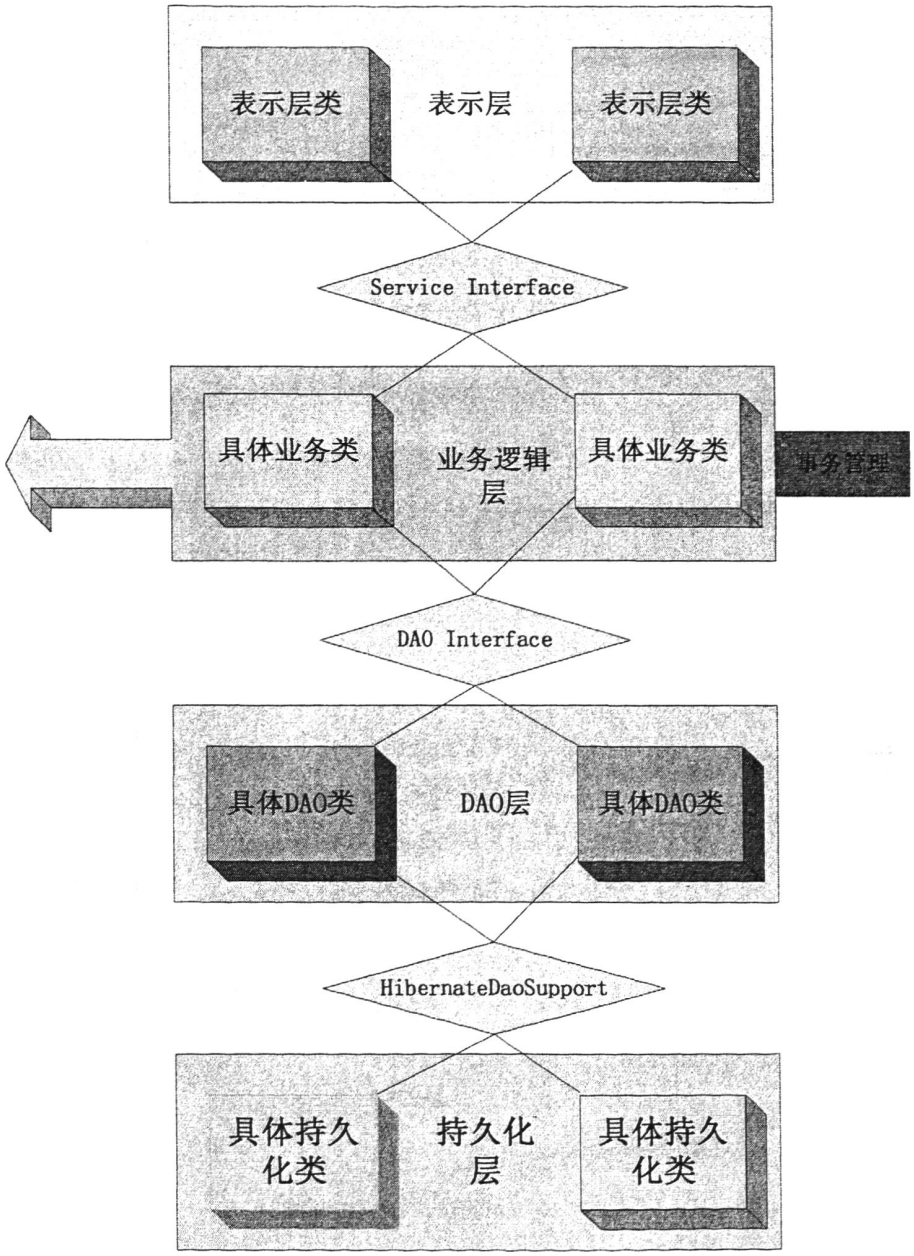

图 4　4 引入了切面管理事务的系统架构图

diagram 4-4 the system architecture diagram of introduce AOP

4.2. 系统功能模块设计

在本文第 3 章中通过对 CMMB 紧急广播信息发布系统的建设目标进行分析确定 CMMB 紧急广播信息发布系统中应该包括对广播消息管理、广播区域管理、网关管理、系统状态监控、日志管理、告警管理、数据维护、账号管理以及对时钟同步管理功能。为了完成以上所述的功能，将整个系统按照功能分为广播消息管理模块、广播区域管理模块、网关管理模块、系统状态监控管理模块、日志管理模块、数据维护模块、账号管理模块以及时钟同步管理模块。

4.2.1. 广播信息管理模块

该部分提供了对 CMMB 紧急广播信息的管理，包括对广播信息的发布、审核、停播、续播、删除的管理。消息的生命期管理包括创建、提交、审核、发送。消息由内容、类型、范围、发送方式、时间、优先级等要素组成。消息的发布只能由消息发布员完成，消息的审核只能有消息审核员来完成，系统管理员不具有消息发布和审核功能。

消息的生命周期是从被创建到最终被发送出去，消息的生命周期管理是系统的核心业务流程。消息的生命周期如 4-5 图所示：

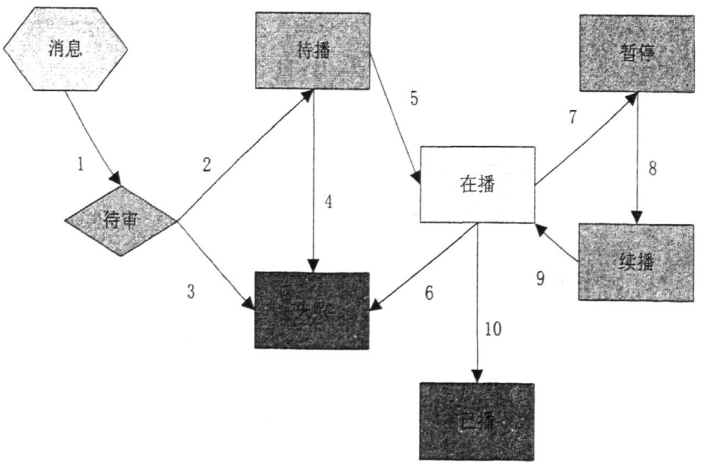

图 4 5 紧急广播消息生命周期

diagram 4-5 Life cycle of the emergency broadcast message

(1)创建消息，消息状态变为待审，等待“消息审核员”审核。

(2) “消息审核员”审核消息通过，状态变为待播，等待发送。

(3) “消息审核员”审核消息未通过，状态变为失败。

(4)等待发送的消息(待播、待审状态)被用户删除或者消息过了发送时间，消息状态变为失败。

(5)发送消息，当消息满足计划条件后被执行，消息状态为在播，等待完成。

(6)正在发送的消息被用户删除或者发送消息出现异常，消息状态变为中止。

(7)正在发送的消息被用户暂停发送，消息状态变成暂停

(8)暂停的消息续播，

(9)消息状态变成在播

(10)完成消息发送，消息状态变为已播。

已播、中止状态的消息可以进行备份并从系统中清除，消息在未审核前可以修改，在其他任何状态下都不能编辑修改，如果需要修改消息只能采取删除后新建的方式来实现。

消息被创建时需要审核才能提交发送，那么消息进入审核流程。流程如图4-6所示：

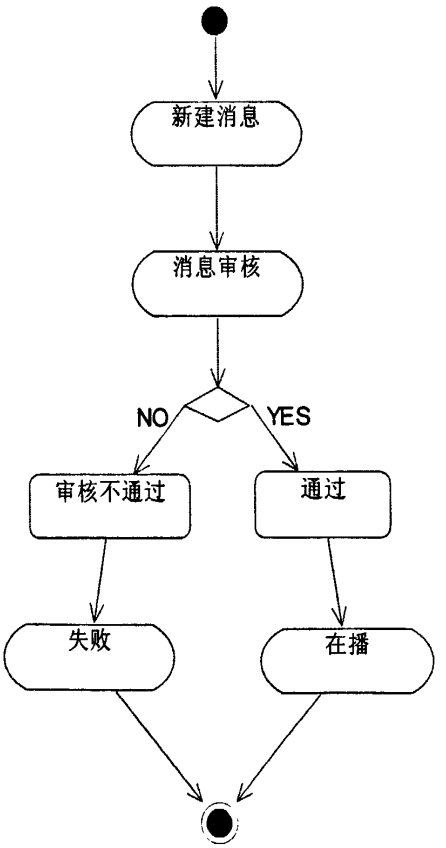

图 4 6 紧急广播消息审核流程

diagram 4-6 approval process for emergency broadcast information

(1)消息被提交审核，进行自动审核。

(2)自动审核不通过，消息提交给“消息审核员”进行人工审核。

(3)自动审核通过，判断消息的关联 CP 是否要求强制审核。

(4)如果要求强制审核，消息提交给“消息审核员”进行人工审核。

(5)如果不需要强制审核，消息被自动设定为审核通过，消息状态未“待播” 。

(5) “消息审核员”人工审核消息，如果审核通过，则消息状态更新为“待播” ；如果不通过，消息状态更新为“中止” 。

广播消息管理用例图如图 4-7 所示：

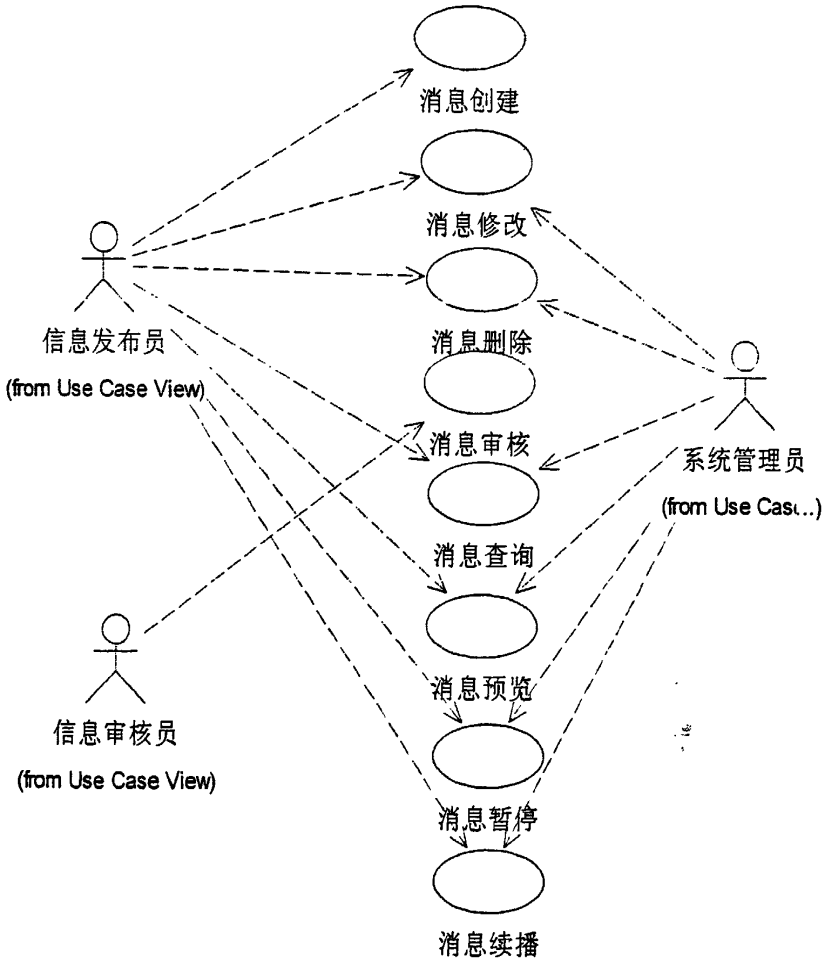

图 4 7广播消息管理用例图

diagram 4-7 the use case diagram of message manage

广播消息管理主界面如 4-8 图所示：

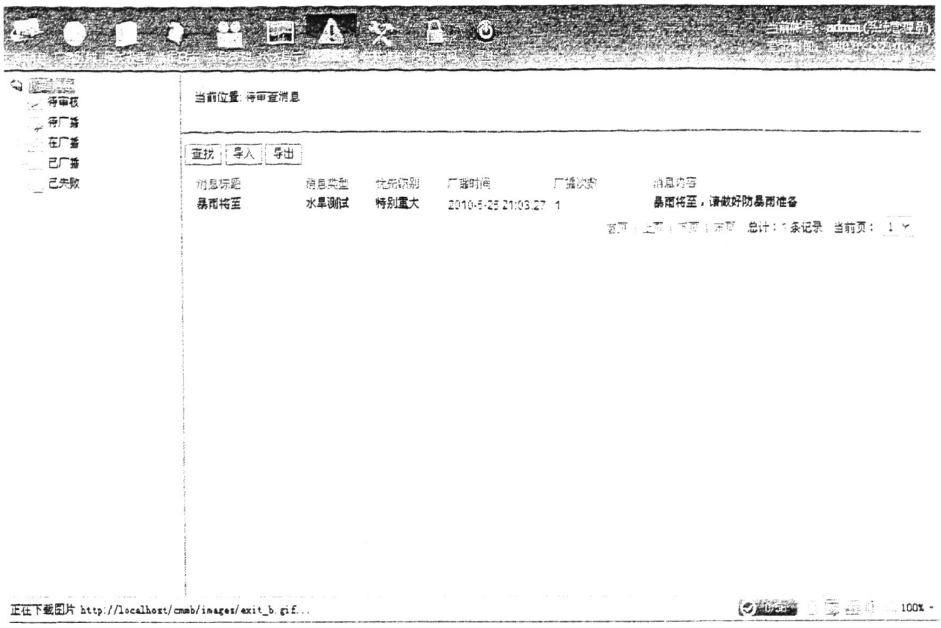

图 4 8 广播消息管理主界面

diagram4-8 the main page of message manage

广播信息添加页面如图 4-9 所示：

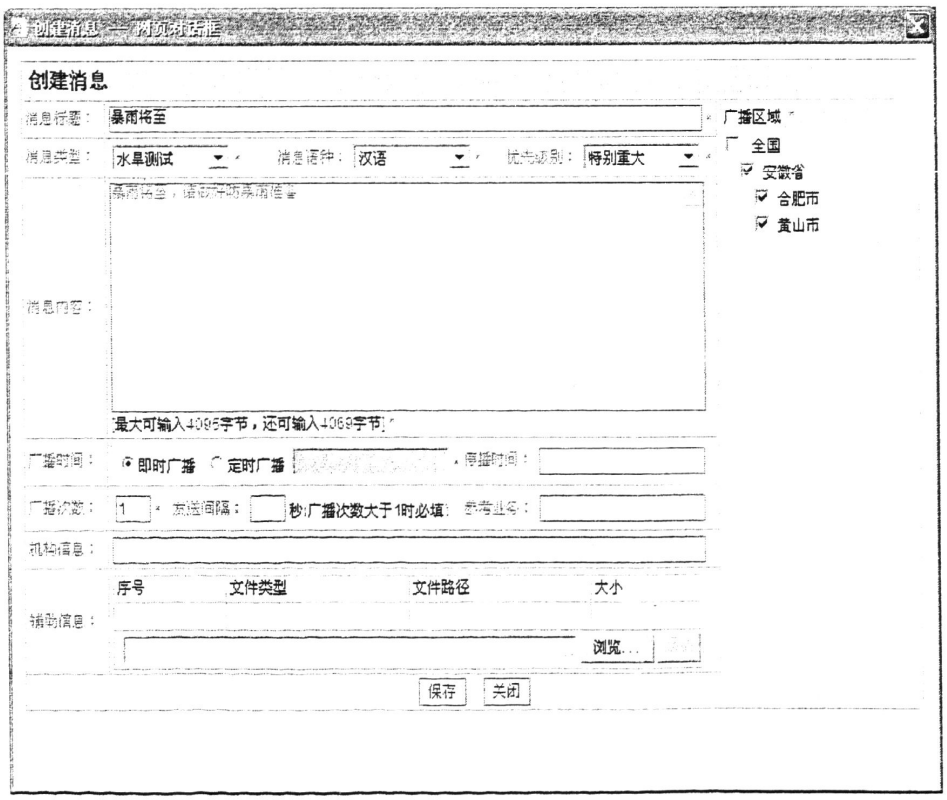

图 4 9 广播信息添加页面

diagram4-9 the page of adding message

广播信息审核页面如图 4-10 所示：

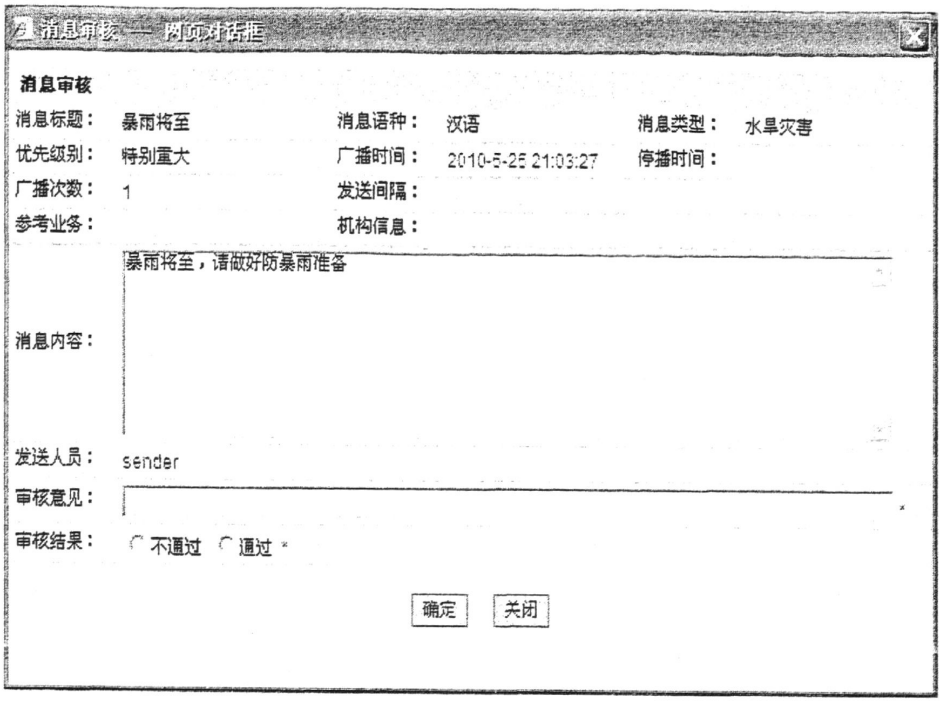

图 4　10 广播信息审核页面

diagram4-10 the page of message audit

4.2.2. 广播区域管理模块

消息区域管理是指对消息发送范围进行管理。主要负责提供信息发布区域的增加、删除、修改和存储等管理，并可以和电子地图相结合提供更加直观的网格化区域的设置，一旦设置好区域地理信息，就可以提供给发布员在录入紧急广播信息时选择相应的广播区域使用。

消息区域管理用例图如图 4-11 所示：

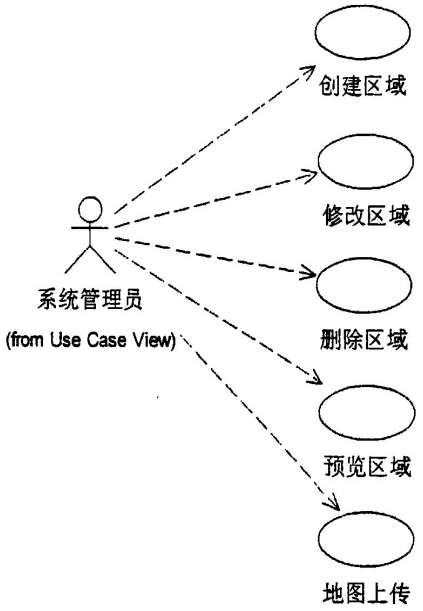

图 4 11 消息区域管理用例图

diagram4-11 use case diagram of area manage

广播区域管理主界面如图 4-12 所示：

图 4 12 广播区域管理主界面

diagram 4-12 the main page of area manage

4.2.3. 网关管理模块

网关管理主要负责提供各种紧急通知发布通道（网关）的配置管理，包括发布通道的增加、删除和修改，通过通道管理就可以在 CMMB 紧急广播发布平台上灵活扩展新的发布网关。

网关管理用例图如图 4-13 所示：

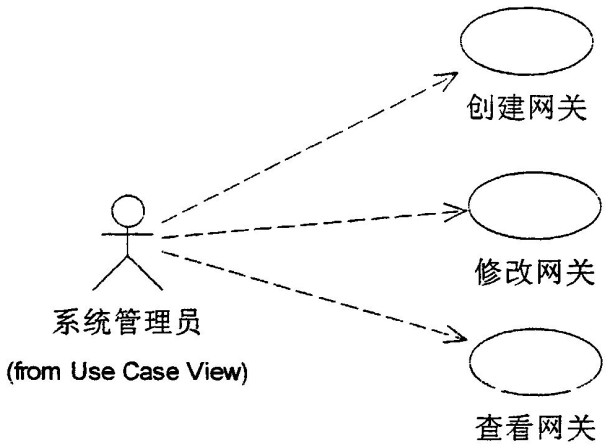

图 4　13 网关管理用例图

diagram4-13 use case diagram of connection manage

网关管理主界面如图 4-14 所示：

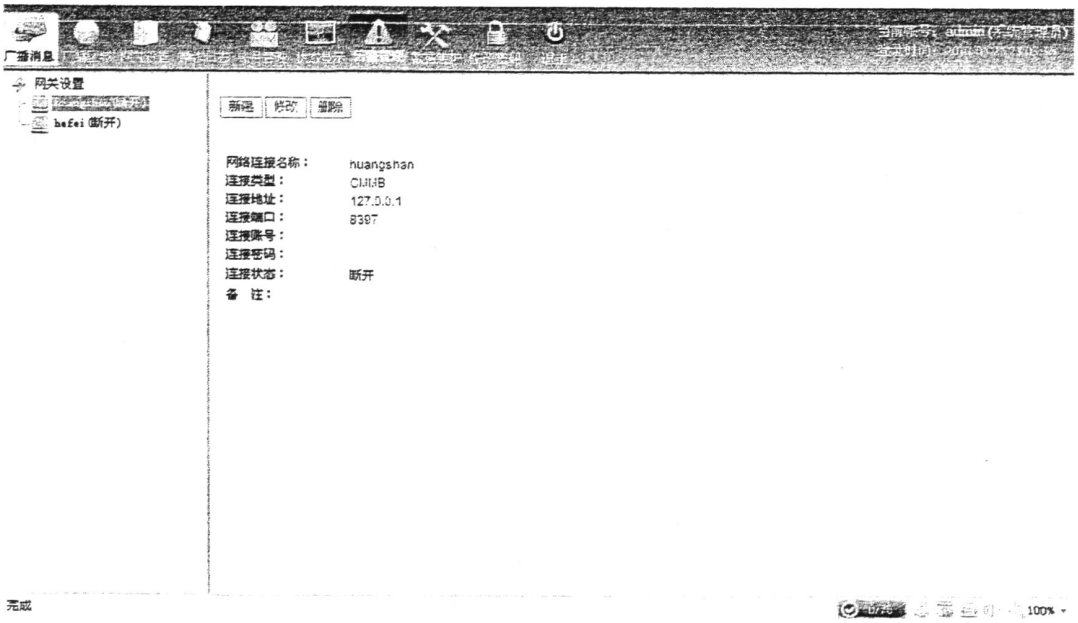

图 4　14 网关管理主界面

diagram4-14 the main page of connection manage

4.2.4. 系统状态监控管理模块

监控管理主要完成对系统 CPU、硬盘、数据库的状态进行监控，能及时反应出系统的运行状态。当超过一定的值的时候就会产生告警。

系统状态监控管理主界面如图 4-15 所示：

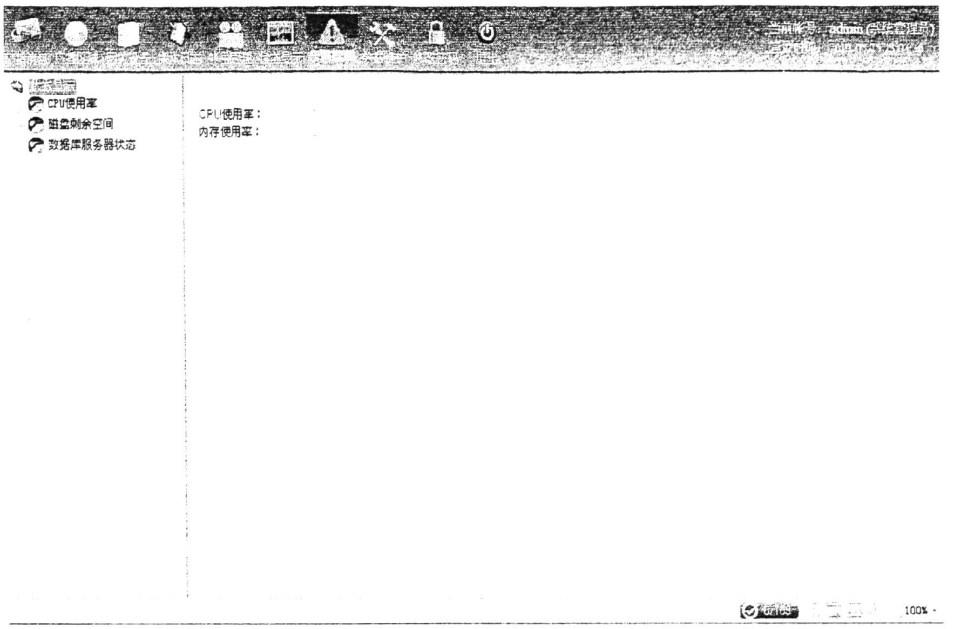

图 4 15 系统状态监控管理主界面

diagram4-15 the main page of system monitor

4.2.5. 日志管理模块

日志管理主要负责完成用户操作记录和系统运行记录，以方便对系统运行的日常跟踪、维护和管理。记录系统产生的事件，主要为人机交互操作。事件分为“系统事件”、“用户事件”两种。需要记录为事件的操作如下：

系统事件：

(1)系统启动、终止

(2)网关的接入、断开

用户事件：

(1)用户登录、注销

(2)删除、新建消息、复制消息、备份消息、审核消息

(3)新增、修改、删除、禁止、启用 CP、配置 CP 业务参数

(4)新增、修改、删除用户，修改当前用户密码

(5)配置区域网格、新建、配置、删除、导出网格地图

(6)新建、修改、删除、复制、移动区域

(7)配置网络连接参数（新建、删除、修改）

(8)消息类型配置（新增、删除、修改）

(9)告警确认、导出

(10)事件导出

日志管理主界面如图 4-16 所示：

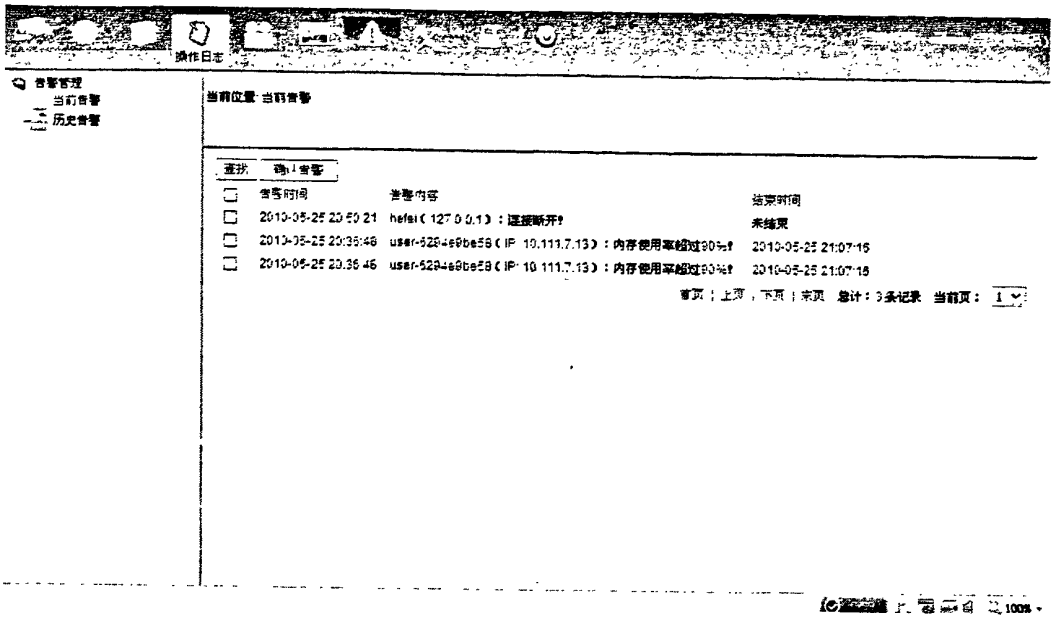

图 4　16 日志管理主界面

diagram4-16 the main page of event manage

4.2.6. 数据维护模块

CMMB 紧急广播信息发布系统中的紧急广播信息以及产生的日志既可以通过手工的方式也可以通过设置系统然后由系统定时去删除的功能。

数据管理主界面如图 4-17 所示：

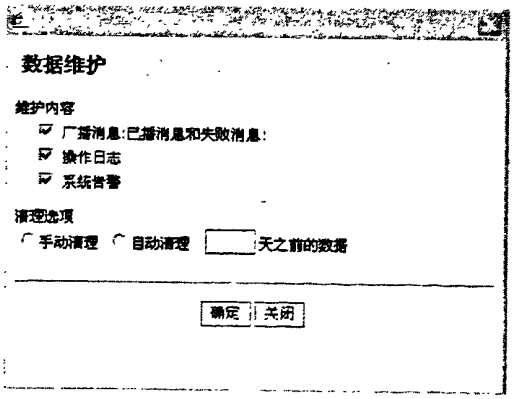

图 4　17 数据管理主界面

diagram4-17 the main page of data maint

4.2.7. 账号管理模块

用户、角色以及权限管理。用户是指可以登录系统进行功能操作的人员。用户拥有不同的角色，系统角色由“超级管理员”、“系统管理员”、“消息审核员”、“消息发布员”组成。每个用户每种角色分别拥有不同的权限。“超级管理员”拥有创建“系统管理员”的权限。本系统中的权限采用 AOP 原理实现。“超级管理员”允许增删改“系统管理员”、“消息审查员”、“消息发布员”“系统管理员”允许增删改“消息审查员”、“消息发布员”。系统管理员除了不能发布紧急广播信息，具有系统中所有的权限。

本系统的权限模块模型图如 4-18 图所示：

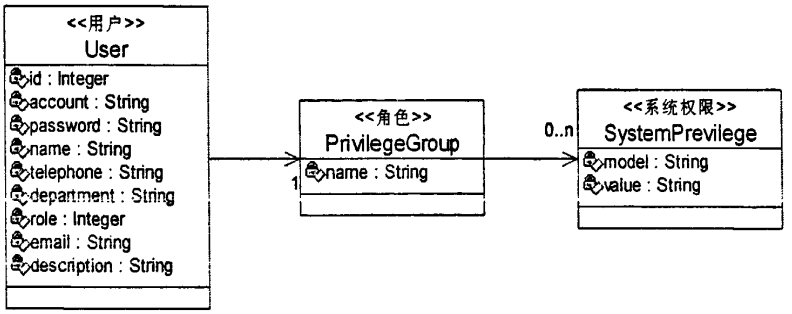

图 4　18 系统权限模块模型图

diagram 4-18 System permissions module model diagram

账号管理模块用例图如 4-19 所示：

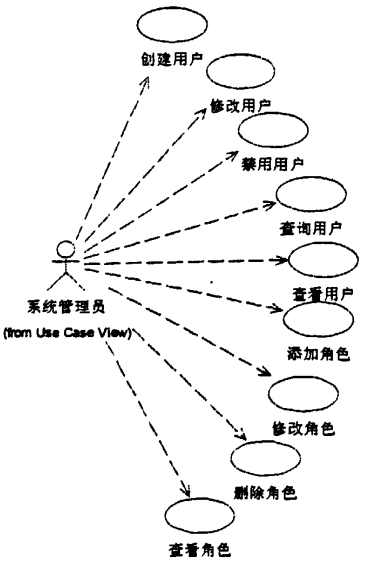

图 4　19 账号管理模块用例图

diagram 4-19 use case diagram of user manage

账号管理主界面如图 4-20 所示：

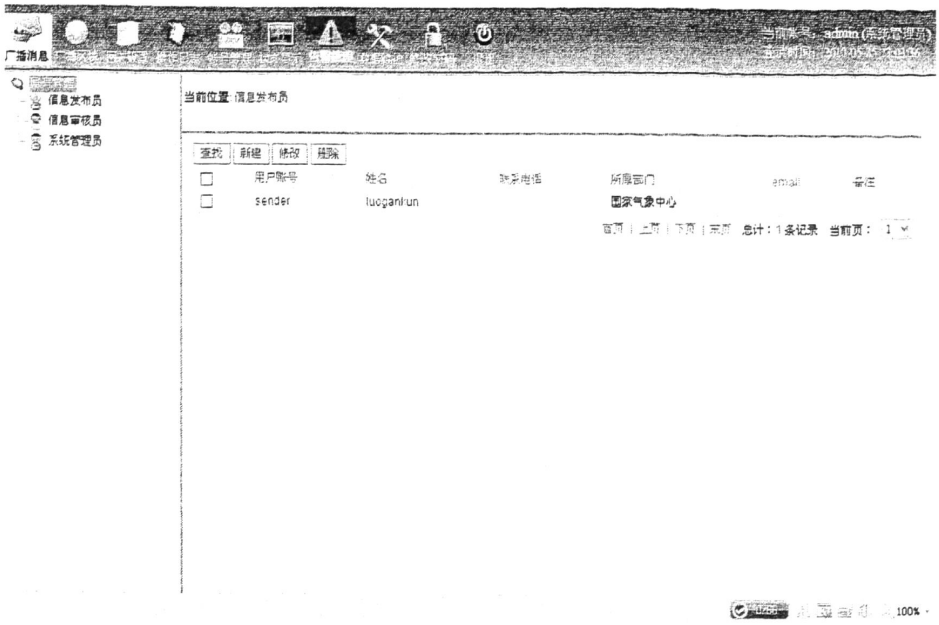

图 4 20 账号管理主界面

diagram4-20 the main page of user manage

4.2.8. 时钟同步管理模块

通过 NTP 协议，与网络时钟服务器建立通信连接，获取时钟信息，并同步调整本机的时钟，使移动多媒体广播 ESG 的各模块能参照网络同步时钟协同工作。

4.3. 领域模型设计

领域模型设计是需求设计的关键步骤。它帮助系统分析人员、用户认识现实业务的工具，描述的是业务中涉及到的实体及其相互之间的关系，它是需求分析的产物，与问题域相关。领域模型是需求分析人员与用户交流的有力工具，是需求分析人员与用户共同理解的概念，是彼此之间交流的语言。而数据模型是系统设计、实现的一部分，描述的是对用户需求在数据结构上的实现，仅此而已。当然数据模型中的概念模型设计与领域模型类似，缺乏的是实体之间更广泛的关系描述。

领域模型设计是系统设计中非常重要的一个环节。领域模型是一切系统设计的基础，通俗的说领域模型设计就像高楼大厦的根基一样，如果设计不合理、不完善，将在系统开发过程中、甚至到后期的系统维护、功能变更和功能扩充时，引起较多问题，严重时甚至要重新设计，重做大量已完成的工作。

领域模型是对信息世界建模，领域模型的表示方法很多，本文中将使用实体一联系(E—R)模型作为描述现实世界的概念模型，实体一概念数据模型基于对现

实世界的这样一种认识：世界是由一组称为实体的基本对象及这些对象间的联系组成。实体一联系数据模型中包含三个主要的概念：实体集、关系集和属性。

本节将针对用户的需求来完成实体集和联系集的设计并在最后给出系统的E—R 图。

4.3.1. 实体集设计

在 CMMB 紧急广播信息发布系统中紧急广播信息包括紧急广播信息和已播信息两大类，在紧急广播信息中根据状态分为待播、待审、在播等不同状态的广播信息：

(1)紧急广播信息实体：具有广播信息的 ID、类型、名称、内容、语种、网络级别、网络 ID、信息发送的优先级、字符集编码、发送目的地、审核时间、审核人、创建人、创建时间、审核意见、审核结果、附加信息、参考信息、失败原因、广播时间、广播次数、广播间隔以及信息状态等属性。

(2)已播广播信息实体：已播区域、失败区域以及紧急广播信息实体对象中的所有属性。

在 CMMB 紧急广播信息发布系统中采用基于角色的访问控制策略，在本系统中角色指的是对系统中资源的访问权限的集合。在 CMMB 紧急广播信息发布系统中包括系统管理员、信息发布员、信息审核员这三种角色。当为用户分配了角色后，该用户就具有了该角色所具有的访问权限集合。在系统中需要对用户信息、角色信息、访问权限信息进行数据记录，因此需要用户、角色、访问权限三个数据实体。

(3)用户实体：具有用户登录系统时所用的 ID、用户的密码、用户真实姓名、用户的电话号码、用户的部门和用户的 email 地址以及用户的描述等属性。

(4)角色数据实体：具有角色的名称和角色的描述等属性。

(5)访问权限数据实体：具有资源和对资源的访问权限等属性。

CMMB 紧急广播信息发布系统中的紧急广播信息发布区域包括对发布区域的添加、删除、修改、查询以及区域地图上传等多个功能。因此需要紧急广播区域数据实体：

(6)广播区域实体：有广播区域 ID、区域名称、区域描述、区域地图名称等属性。

CMMB 紧急广播信息发布系统中具有很好的日志功能，对系统的操作均会记录在日志中，需要日志操作数据实体：

(7)操作日志数据实体：具有日志 ID、日志类型、产生时间、事件描述、日志产生的账号、等属性。

CMMB 紧急广播信息发布系统对紧急广播信息的发送是通过接入的合法的

网关来进行发送的，需要网关数据实体：

(8) 网关数据实体：具有网关 ID、名称、类型、IP 地址、端口号、接入账号、接入密码、网关描述、网关是否连上标示等属性。

CMMB 紧急广播信息发布系统具有较好的系统监控环境，能够对系统的 CPU、硬盘等物理对象进行监控，因此需要监控数据实体：

(9) 监控数据实体：监控的服务器名称、服务器地址、服务器状态、CPU 使用了、内存使用率、硬盘使用空间、硬盘总空间等属性。

CMMB 紧急广播信息发布系统在发送状况的时候会有很多的告警功能，如：在 CPU 或者内存超过一定的使用率的时候就会参数告警，告知用户系统存在崩溃的危险，在网关在非正常情况下断开的时候也会产生告警等，因此需要告警数据实体：

(10) 告警数据实体：具有告警 ID、告警描述、告警类型、产生告警的主机 IP 地址、告警开始时间、告警结束时间、告警级别、告警确认时间、确认告警人、告警状态、告警确认描述等属性。

CMMB 紧急广播信息发布系统中的紧急广播信息以及产生的日志既可以通过手工的方式也可以通过设置系统然后由系统定时去删除的功能，所以还需要系统数据维护实体：

(11) 系统数据维护实体：具有系统数据维护 ID、手动清除还是自动清除的标志、定时清除数据的间隔数等属性。

4.3.2. 实体关系设计

根据上一节中的分析，将定义如下实体关系。并对不易理解的地方举例说明。

(1)下级区域—上级区域：具有子区域和上级区域实体之间的关系；其映射关系表明了一个上级区域包括多个子区域，而一个子区域只属于一个上级区域

(2)紧急广播消息—区域：在建立紧急广播信息的时候要选择所要发送的区域；其映射关系表明了一个紧急广播消息对应多个信息发布区域。

(3) 区域—网关：系统中的区域与网关一一对应，在新建紧急通知的时候要选择发送的区域，而在真正发送消息的时候是通过与该区域所对应的网关来进行发送的。

(4)用户—角色：是系统互用和系统角色之间的一对多联系集；其映射关系表明了一个用户可以具有多个角色。

(5)角色—权限：是角色和系统权限之间的一对多联系集；其映射关系表明了一个角色具有多个系统权限。

4.3.3. 系统 E-R 图

下面给出整个系统的实体-关系模型图：

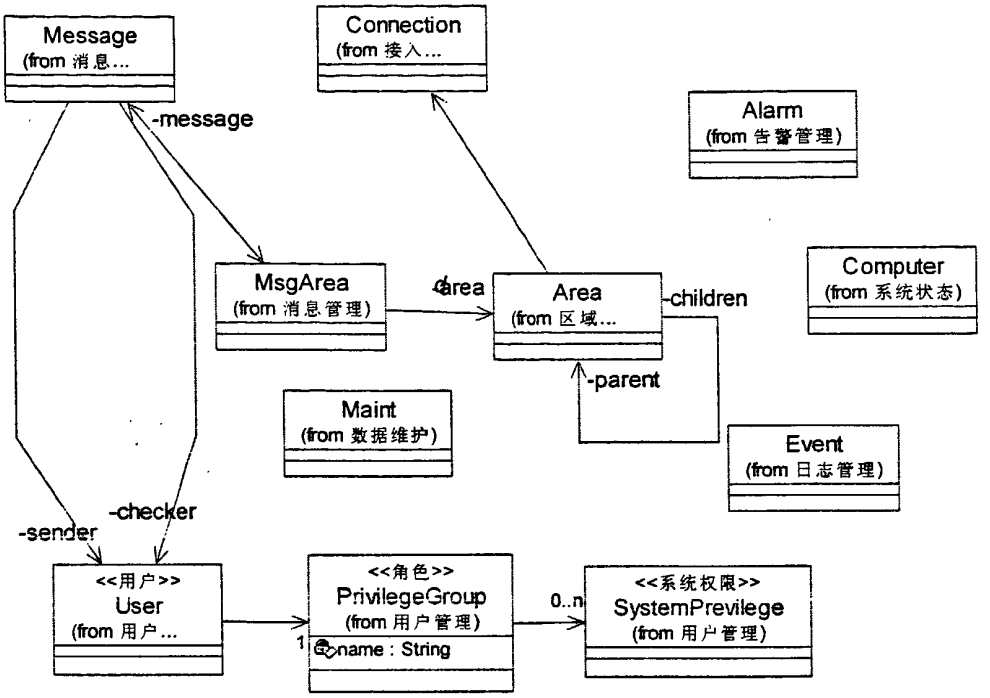

图 4 21 系统 ER 图

diagram 4-21 the system er diagram

5.系统实现

在本文第四章中介绍了CMMB紧急广播信息发布系统架构的设计，在架构设计过程中，将CMMB紧急广播信息发布系统中具有相同职能的类聚集在一起，多种不同职能的类的聚集就产生了层的概念。CMMB紧急广播信息发布系统中具体功能的实现过程也是按照实现底层子系统，通过底层子系统为上层子系统提供功能来逐级实现的。

在CMMB紧急广播信息发布系统中，分为四个层次：数据持久层、DAO层、业务逻辑层和表示接口层。数据持久层负责系统中业务对象的持久化工作，DAO层负责对数据访问对象的细节的工作，业务逻辑层处理短信服务平台的业务逻辑，并调用DAO层完成业务逻辑处理过程中需要的持久化数据。表示接口层负责和用户或者其他系统进行交互，从用户界面或者其他系统获得业务逻辑层工作需要的数据，并通过对业务逻辑层的调用完成业务逻辑处理。在CMMB紧急广播信息发布系统中采用了基于J2EE技术架构的Struts2+Spring+Hibernate组合应用框架来实现上述的软件架构。使用Struts2实现控制层，利用Struts2提供的MVC模型来实现用户界面和流程控制之间的解耦合。使用Hibernate来处理对象关系映射，解决持久性对象的问题。利用Spring提供的IoC来解耦合业务对象。

因为在层次结构中应该遵循下层为上层提供服务，上层对下层一无所知的原则，本节的叙述结构也按照从下层到上层来组织。首先介绍在CMMB紧急广播信息发布系统中如何通过配置文件来将Struts2+Spring+Hibernate三个框架整合起来，然后介绍CMMB紧急广播信息发布系统中数据持久层的实现，在此基础上介绍CMMB紧急广播信息发布系统中业务逻辑层的实现，最后介绍表示接口层的实现。

5.1. 数据持久层实现

在CMMB紧急广播信息发布系统中包含数量众多的数据，例如进行区域管理的数据，用来进行紧急广播信息发送的信息等。系统管理员进行系统管理时系统都会根据已有的区域信息将生成区域树，在信息发送员登陆系统后，系统将根据区域信息生成紧急广播信息发送区域树。在CMMB紧急广播信息发布系统采用面向对象的开发方法.。业务对象的持久化最终必须通过数据访问代码来实现。本文所介绍的CMMB紧急广播信息发布系统采用Hibernate框架来完成业务对象的持久化。下面将以紧急通知信息对象和消息发布区域对象举例来展示如何对业

务对象进行持久化。

5.1.1. 建立对象模型

在 CMMB 紧急广播信息发布系统中，一条紧急广播信息可以发送到多个区域，一个区域可以被发送多条紧急广播信息，所以紧急广播信息和发送区域是一种单向多对多关联关系。在具体实现时在 Message 和 Area 对象之间建立一个中间对象用于记录一条紧急广播信息所对应的要发送的区域。下图是 Message 与 Area 的关联对象模型。

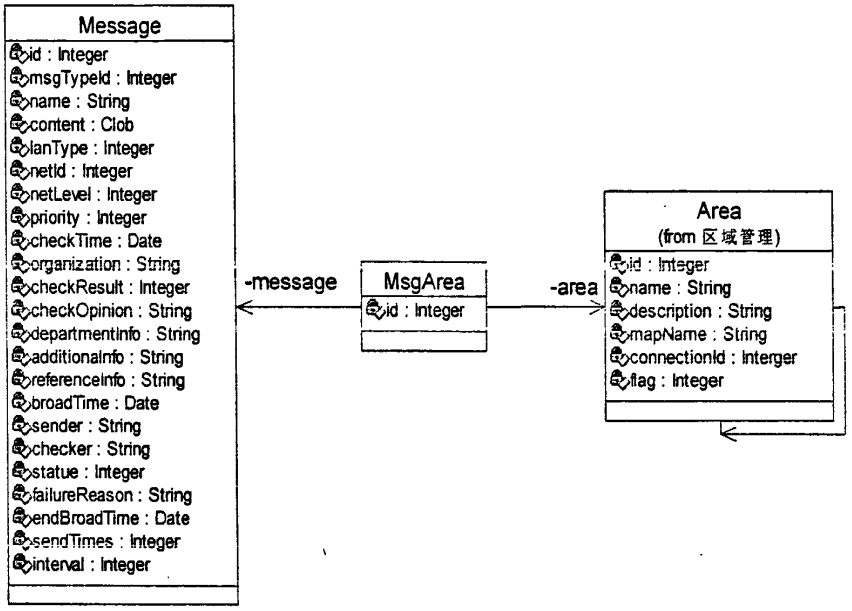

图 5 1 Message 与 Area 的关联对象模型

diagram 5-1 the model of Message and Area and MsgArea

由于在业务对象类中只是定义了业务对象的属性和属性的赋值方法和获取属性值的方法，在这里不再给出具体的 Java 代码。

5.1.2. 建立对象映射文件

在前面部分已经建立了业务对象 Message 以及 Area 之间的关系，下面将使用 Hibernate 来完成对象之间的映射。在 Hibernate 中使用 xml 格式的文件来指定对象和关系数据库之间的映射。在系统运行时，Hibernate 将根据配置文件中的 hibernate. dialect 属性来生产各种不同数据库的 SQL 语句。可以使用单独的配置文件来完成多个业务对象和多个数据库表之间的映射，在这里为每个业务对象建立一个配置文件。Message.hbm.xml 在业务对象紧急广播信息和数据库表 cmmb_message 之间建立对象关系映射。Area.hbm.xml 在业务对象广播区域和数

据库表 cmmb_area 之间建立对象关系映射。MsgArea.hbm.xml 在业务对象广播区域和数据库表 cmmb_msg_area 之间建立对象关系映射。Message.hbm.xml 文件如下表所示：

表 5　1 Message.hbm.xml 配置文件

table5-1 Message.hbm.xml configuration file

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse Persistence Tools
-->
<hibernate-mapping>
    <class name="com.magnus.cmmb.message.pojo.Message"
table="cmmb_message" catalog="cmmb">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="native" />
        </id>
        <property name="msgTypeId" type="integer">
            <column name="msgTypeId_"/>
        </property>
        <property name="name" type="string">
            <column name="name_" length="100"   />
        </property>
        <property name="content" type="string">
            <column name="content_"/>
        </property>
        <property name="lanType" type="integer">
            <column name="lanType_"/>
        </property>
        <property name="netId" type="integer">
            <column name="netId_"/>
        </property>
```

```
<property name="netLevel" type="integer">
    <column name="netLevel_"/>
</property>
<property name="priority" type="integer">
    <column name="priority_"/>
</property>
<property name="codeScheme" type="integer">
    <column name="codeScheme_"/>
</property>
<property name="destination" type="integer">
    <column name="destination_"/>
</property>
<property name="checkTime" type="timestamp">
    <column name="checkTime_"/>
</property>
<property name="checkResult" type="integer">
    <column name="checkResult_"/>
</property>
 <property name="checkOpinion" type="string">
    <column name="checkOpinion_" length="100"   />
</property>
  <property name="failureReason" type="string">
    <column name="failureReason_" length="400"   />
</property>
 <property name="departmentInfo" type="string">
    <column name="departmentInfo_" length="100"   />
</property>
 <property name="additionalInfo" type="string">
    <column name="additionalInfo_" length="100"   />
</property>
 <property name="referenceInfo" type="string">
    <column name="referenceInfo_" length="100"   />
</property>
  <property name="broadTime" type="timestamp">
```

```
            <column name="broadTime_"/>
        </property>
        <property name="sendTimes" type="integer">
            <column name="sendTimes_"/>
        </property>
          <property name="endBroadTime" type="timestamp">
            <column name="endBroadTime_"/>
        </property>
         <property name="sender" type="string">
            <column name="sender_" length="20"/>
        </property>
        <property name="checker" type="string">
            <column name="checker_" length="20"/>
        </property>
        <property name="status" type="integer">
            <column name="status_"/>
        </property>
       <property name="interval" type="java.lang.Integer">
          <column name="interval_" />
       </property>
        <property name="organization" type="string">
            <column name="organization_" length="128"/>
        </property>
    </class>
</hibernate-mapping>
```

Area.hbm.xml 配置文件如下表所示：

表 5　2 Area.hbm.xml 配置文件

table5-2 Area.hbm.xml confugration file

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse Persistence Tools
-->
```

```
<hibernate-mapping>
    <class name="com.magnus.cmmb.area.pojo.Area" table="cmmb_area"
catalog="cmmb">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="native" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="name_" length="20"    />
        </property>
        <property name="description" type="java.lang.String">
            <column name="description_" length="200" />
        </property>
         //多对一关联映射
        <many-to-one name="parent" column="pid_"
                    class="com.magnus.cmmb.area.pojo.Area"/>
         //一对多关联映射
        <set name="children" inverse="true" lazy="false" cascade="all"
            order-by="id">
          <key column="pid_"/>
          <one-to-many class="com.magnus.cmmb.area.pojo.Area"/>
        </set>
         <property name="mapName" type="java.lang.String">
            <column name="map_name_" length="2000"    />
        </property>
         <property name="connectionId" type="java.lang.Integer">
            <column name="connection_id_"/>
        </property>
        <property name="flag" type="integer">
          <column name="flag_"/>
        </property>
    </class>
</hibernate-mapping>
```

MsgArea.hbm.xml 配置文件如下表所示：

表 5　3 MsgArea.hbm.xml 配置文件

table5-3 MsgArea.hbm.xml configuration file

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse Persistence Tools
-->
<hibernate-mapping>
    <class name="com.magnus.cmmb.message.pojo.MsgArea"
table="cmmb_msg_area" catalog="cmmb">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="native" />
        </id>
        <property name="msgId" type="integer">
            <column name="msgid_"/>
        </property>
        <property name="areaId" type="integer">
            <column name="areaid_"    />
        </property>
    </class>
</hibernate-mapping>
```

Hibernate 可以通过以上的配置信息完成业务对象和数据库关系之何的对应映射关系。<class>元素指定了 Message 类和数据库麦 cmmb_message 进行对应，<id>元素指明了对象标识和数据库主键之间的对应，<property>元素指明了业务对象中的属性和数据库中字段之间的对应，<set>元素指明了在区域对象中包含多个子一曲对象，并指明了在 cmmb_area 表中。

5.1.3. 建立 BaseDAO

上文已经介绍了建立业务对象并通过配置文件完成了业务对象和数据库关系之间的映射。接下来将建立用来完成业务持久化任务的类。在这里使用数据访

问对象(DAO)把数据访问特定的代码与应用程序的其余部分分离开来。首先定义一个通用的 BaseDAO，CMMB 紧急广播信息发布系统中所有的对象的 DAO 均继承自 BaseDAO，在 BaseDAO 中定义了对象常见的方法(如:增删改查操作等)。这样的好处是当需要对业务对象进行持久化时，具体调用类 DAO 只需要继承 BaseDAO 提供了哪些方法就可以了。而不需要了解具体的实现是使用 JDBC 还是 HIBERNATE 等具体技术，从而达到解耦的效果。

5.1.4. 建立对象 DAO 的具体实现

在上一节介绍了对业务对象 DAO 进行操作的持久化通用类 BaseDAO，下面将建立系统中具体对象的 DAO 类的具体实现。在 CMMB 紧急广播信息发布系统中采用 Hibernate 框架作为持久层的实现。

要对数据库进行操作首先要获得数据库连接，在 Hibernate 中可以通过在持久化类中加入 SessionFactory 类型的属性来获得数据库连接，在这里 SessionFactory 是一个重量级的对象，在使用时会耗费大量的资源，所以在应用中应该使用单例模式来控制生成 SessionFactory 对象。通过 SessionFactory 的 getSession()方法可以获得一个轻量级的数据库连接，可以使用这个连接来进行数据库的操作。在 CMMB 紧急广播信息发布系统中并不直接使用 Hibernate 框架中提供的 getSession()方法来获得数据库连接，而是使 BaseDAO 继承 Spring 框架中提供的 HibemateTemplate 类来简化实现过程。在 HibemateTemplate 类中已经包含了 SessionFactory 属性，并提供了对业务对象持久化操作的一些方法。下面给出 MessageDAO 的类图，而不给出具体的代码。

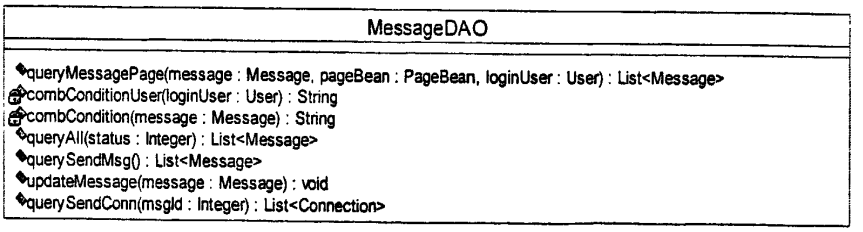

图 5 2 MessageDAO 类图

diagram 5-2 the class diagram of Message class

5.2. 业务逻辑层实现

CMMB 紧急广播信息发布系统的业务逻辑层处理紧急广播信息的实际业务需求。与数据持久层相比，业务逻辑层具有更多的灵活性。本节将从定义业务逻辑层边界、利用 IoC 降低业务层对象之间耦合、定义声明式事务管理几个方面来介绍 CMMB 紧急广播信息发布系统业务逻辑层的实现。

5.2.1. 定义业务逻辑层边界

这里提到的业务逻辑层边界，是指业务逻辑层所要完成的功能集合。相对于CMMB 紧急广播信息发布系统中功能的具体实现，系统中所具有的功能更具有稳定性，因此可以给系统定义出一个一劳永逸、不再更改的抽象设计，此设计允许有无穷无尽的行为在实现层被实现。通过业务逻辑边界的定义，使得所提供的服务能满足表示层的需求，使得表示层不需要跨越业务逻辑层去访问数据持久层，避免了持久化代码分散在表示层中，提高了系统的可维护性以及降低系统的耦合性。相对于具体实现而言，接口是较稳固的，在 CMMB 紧急广播信息发布系统的业务逻辑层中，为每个具体功能建立接口，例如在系统管理部分为用户管理、区域管理等子功能建立 UserService 接口和 AreaService 接口。在 CMMB 紧急广播信息发布系统持久层的实现中提到了应用 Facade 模式来降低业务逻辑层对数据持久层访问的复杂性，在表示层和业务逻辑层之间存在同样的问题，在这里同样使用 Facade 模式来降低表示层和业务逻辑层之间的访问复杂性。在业务逻辑层中通过接口来访问 DAO 层，并不关心真正的 DAO 实现，这样 Service 层就向一个大管家一样，在 Service 接口中定义了获得实现该功能模块服务类的方法。下面给出了表示层(XXXAction)与业务逻辑层(XXXService)以及业务逻辑层XXXService)与 DAO(XXXDAO)层的解耦关系图，XXXAction 只能调用 XXXService 的方法，XXXService 只能调用 XXXDAO 的方法，它们之间不能跨层调用，从而达到解耦的目的。

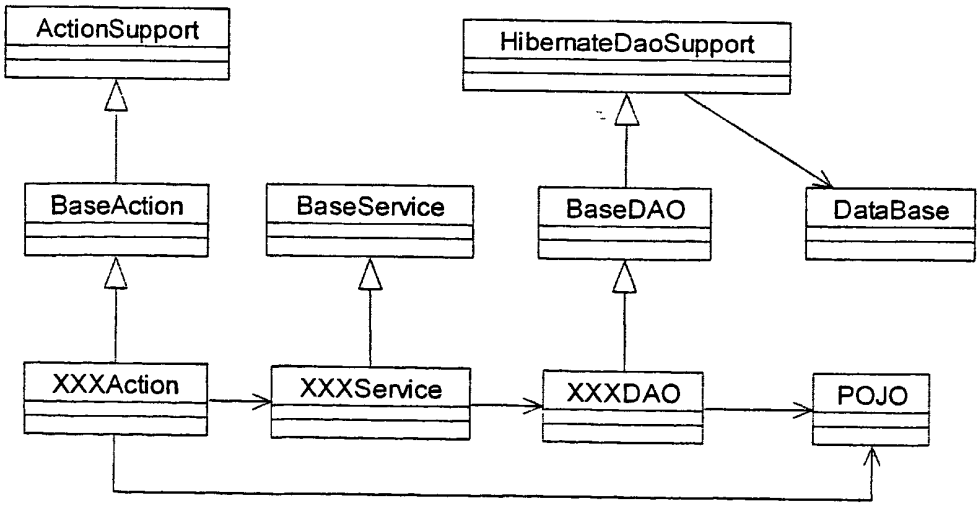

图 5　3 系统各层次之间的调用图

diagram 5-3 System calls to map between different levels

5.2.2. 利用 IoC 降低业务层对象之间的耦合

在上一节中通过一个抽象的层次定义了业务逻辑层的边界，可以看到在应用了面向接口的编程后，将具体实现隐藏在接口下面，这样具体实现类的替换不会影响到引用类。在传统的编程方式中大多采用直接指定具体类的信息或者使用工厂方法来获得具体类的实例。如果直接指定具体类的信息，在具体类发生变化的时候必须要通过修改具体的代码才能进行重构，很不利于开发和维护；如果使用工厂方法来获得具体类的实例的却通过修改配置文件而并不需要修改代码的方式达到具体类的更换，但是在一个大型的系统中使用这种方式的话，会出现很多的工厂，进而导致工厂泛滥。在 CMMB 紧急广播信息发布系统中利用 Spring 框架中提供的 IoC 容器来解决以上提到的问题。Spring 中是通过在配置文件中定义对象之间的依赖关系，对象与对象之间的依赖对于编程人员来说是可配置的，这样就避免了对代码的修改，提高了代码的可重用性并且不需要自己提供很多的工厂。下面使用实例来说明在 CMMB 紧急广播信息发布系统中是如何使用 IoC 来降低业务逻辑层对象之间的耦合性。

表 5　4 MessageService.java 部分代码

table5-4 the code of MessageService.java

```
public class MessageService extends BaseService<Message> {
    private MessageDAO messageDAO;
    private MsgAreaDAO msgAreaDAO;
    private EndPlayMessageDAO endPlayMessageDAO;
    private AreaDAO areaDAO;
    public AreaDAO getAreaDAO() {
        return areaDAO;
    }
    public void setAreaDAO(AreaDAO areaDAO) {
        this.areaDAO = areaDAO;
    }
    @SuppressWarnings("unchecked")
    @Override
    protected BaseDAO getBaseDAO() {
        return messageDAO;
    }
    public MessageDAO getMessageDAO() {
        return messageDAO;
```

```
    }
    public void setMessageDAO(MessageDAO messageDAO) {
        this.messageDAO = messageDAO;
    }
}
```

以上表是 MessageService.java 类中的部分代码，可以看到为了实现 getMessageDAO()方法，并没有指定返回的 MessageDAO 的具体实现类，而是通过 Spring 的 IoC 来注入真正的实现类。Spring 的配置文件如下：

表 5　5 MessageService 类与 MessageDAO 类解耦的 Spring 配置文件

table5-5 MessageService and MessageDAO in the spring configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"
"http://www.springframework.org/dtd/spring-beans-2.0.dtd">
<beans default-autowire="byName">
    <bean id="areaService" parent="baseTxService">
        <property name="target">
            <bean class="com.magnus.cmmb.area.service.AreaService"/>
        </property>
    </bean>
    <bean id="messageService" parent="baseTxService">
        <property name="target">
            <bean class="com.magnus.cmmb.message.service.MessageService"/>
        </property>
    </bean>
    <bean id="msgAreaService" parent="baseTxService">
        <property name="target">
            <bean class="com.magnus.cmmb.message.service.MsgAreaService"/>
        </property>
    </bean>
</beans>
```

上面是通过配置文件的方式将具体的实现类配置在一个文件中，这样要是想修改某个类的实现类就只需要在这个配置文件中修改即可，并不需要去修改代码，大大降低了代码的耦合程度。

5.2.3. 定义声明式事务管理

在软件中，业务层的一个操作要么全部成功要么全部失败的操作成为事务。事务允许你把几个操作组成一个单一的工作单元，这个工作单元要么全部发生要么全部不发生。

下面仍然以紧急广播信息管理功能为例解释在 CMMB 紧急广播信息发布系统中如何使用声明式事务管理。在信息管理部分，包括查看紧急广播信息信息，修改紧急广播信息，删除紧急广播信息等功能。在删除紧急广播信息的操作时，既要删除广播信息表，又要删除该条广播信息对应的发送区域即 cmmb_msg_area 中的数据，这就需要同时删除两个表内的数据，这样才能保证数据库的完整性。下表给出配置声明式事务处理的部分配置文件。

表 5 6 事务在 Spring 文件中的配置

table5-6 the configuration of the transaction in Spring configuration file

```
<bean id="baseTxService"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean"
abstract="true">
        <property name="transactionAttributes">
            <props>
                <prop key="query*">PROPAGATION_REQUIRED</prop>
                <prop key="select*">PROPAGATION_REQUIRED,readOnly
                </prop>
                <prop key="find*">PROPAGATION_REQUIRED,readOnly
                </prop>
                <prop key="view*">PROPAGATION_REQUIRED,readOnly
                </prop>
                <prop key="is*">PROPAGATION_REQUIRED</prop>
                <prop key="monitor*">PROPAGATION_REQUIRED</prop>
                <prop key="log*">PROPAGATION_REQUIRED</prop>
                <prop key="get*">PROPAGATION_REQUIRED</prop>
                <prop key="save*">PROPAGATION_REQUIRED</prop>
                <prop key="add*">PROPAGATION_REQUIRED</prop>
                <prop key="edit*">PROPAGATION_REQUIRED</prop>
                <prop key="update*">PROPAGATION_REQUIRED</prop>
                <prop key="delete*">PROPAGATION_REQUIRED</prop>
                <prop key="check*">PROPAGATION_REQUIRED</prop>
```

```
            </props>
        </property>
    </bean>
```

在上面的配置文件中指明使用了 TransactionProxyFactoryBean 类对 Service 类中的以“delete”为前缀的方法使用 PROPAGATION_REQUIRED 方式的事务管理。

5.3. 展现层实现

表示层主要用来与用户进行交互，从用户界面获得输入经过处理后输出处理结果。在 CMMB 紧急广播信息发布系统中使用 Struts2 框架作为表示层的主要实现技术。在 CMMB 紧急广播信息发布系统中，使用前面介绍过的领域模型作为 Strut2 应用中的模型，使用 Struts2 提供的标签实现视图，通过继承 Struts 中提供的 ActionSupport 类来实现控制器。下面将以为系统中添加新信息为例来介绍视图的实现、控制器的实现。

5.3.1. 页面视图的实现

在 CMMB 紧急广播信息发布系统中新建紧急广播信息使用 MessageAction.java 类作为页面表单类，需要 Message 类、Area 类以及 MsgArea 类来完成一般只需要一些赋值和取值操作，在这里不给出具体的代码。定义了表单类后，当页面请求到 MessageAction.java 中的 doAdd()方法的时候 struts2 会可以自动到页面中收集要提交的数据，然后执行 doAdd 中的方法并跳转到指定的页面中去。Struts 配置文件的部分代码如下表所示。

表 5 7 MessageAction 的 struts 配置文件

table5-7 the configuration of MessageAction in struts.xml

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.1.1//EN"
"http://www.opensymphony.com/xwork/xwork-1.1.1.dtd">
<xwork>
    <package name="message" extends="default">
        <action name="message" class="messageAction">
            <result name="preAdd" type="dispatcher">
                /jsp/message/addMessage.jsp
            </result>
            <result name="doAdd" type="dispatcher">
```

```
        /jsp/message/addMessage.jsp
    </result>
    <result name="preEdit" type="dispatcher">
        /jsp/message/editMessage.jsp
    </result>
    <result name="doEdit" type="dispatcher">
        /jsp/message/editMessage.jsp
    </result>
    <result name="doView" type="dispatcher">
        /jsp/message/showMessage.jsp
    </result>
    <result name="doQuery" type="dispatcher">
        /jsp/message/toAuditMsgList.jsp
    </result>
    <result name="doQueryAll" type="dispatcher">
        /jsp/message/allMsgList.jsp
    </result>
    <result name="preCheck" type="dispatcher">
        /jsp/message/checkMessage.jsp
    </result>
    <result name="doCheck" type="dispatcher">
        /jsp/message/checkMessage.jsp
    </result
    <result name="importMsg" type="dispatcher">
        /jsp/message/importInput.jsp
    </result>
    <result name="importResult" type="dispatcher">
        /jsp/message/importInput.jsp
    </result>
    <result name="downLoad" type="stream">
            <param name="contentType">application/xml</param>
            <param name="contentDisposition">
                attachment;filename=${fileName}.xml
            </param>
```

```
                    <param name="inputName">downloadFile</param>
            </result>
            <interceptor-ref name="myStack"/>
              <interceptor-ref name="fileUploadStack"/>
        </action>
    </package>
</xwork>
```

具体的页面采用 struts2 的标签来实现，因为比较简单且这里不给出具体的代码。

5.3.2. 控制层的实现

Struts2 框架中的对于控制器部分的实现应用了前端控制器模式，从 web 应用的配置文件中可以看到，所有的 URL 都对 org.apache.struts2.dispatcher.FilterDispatcher 类做出请求，而 org.apache.struts2.dispatcher.FilterDispatcher 类通读取 Struts2 的配置文件后来进行页面的跳转。

在 Struts2 框架中提供了 ActionSupport 类来作为具体控制器类的基类，具体的控制器类只需要继承 ActionSupport 类，并覆写父类中的 excute()方法。建立了具体的控制器类后还需要在配置文件中定义 URL 和控制器的映射关系以及控制器处理完请求后页面可能的跳转情况。在这里不再给出 MessageAction 类的代码。

6. 结论和展望

6.1. 结论

气象灾害以及其他公共灾害给人类造成的损失越来越引起社会的广泛关注，提高预警能力，加强预警，将灾害预报的信息最早、最及时传递到需要预警的地区，告知人们做好准备是降低灾害损失的有效途径。

国家广播电影电视总局为提高广播预警能力，以上海为试点，开展了 CMMB 紧急广播信息发布系统建设工作，并成立了项目组进行开发。本论文以该项目为背景。试图建立起基于 CMMB 网络的信息发布系统，在出现国家规定的紧急情况和突发事件时，可以实现向用户提供紧急通告服务，紧急广播以文字信息为基本方式，可扩展支持音频、图片、图像等方式，应实现多语种和多编码，以适应少数民族地区和在境内的境外人员，当前端发送紧急广播消息时，接收终端应能强行切换接收紧急广播消息，并能根据紧急事件的级别和类型采取相应的展示方式以达到预警目的。

本文主要论述了基于 J2EE 架构的 CMMB 紧急广播信息发布系统的设计与实现，其中重点论述了 CMMB 紧急广播信息发布系统中的系统分析、系统设计及系统实现。本着一切从满足用户需求出发，通过对系统的分析寻找用户的确切需求及业务流程，并根据系统分析的结果完成了系统软件架构设计、功能模块设计、数据库总体设计以及系统的基本实现。

CMMB 紧急广播信息发布系统使用了分层架构，在表示层、业务逻辑层、DAO 层以及数据持久层分别应用了 Struts2、Spring 和 Hibernate 技术。其中 Spring 是系统中的核心技术，通过 IOC、AOP 等技术，使系统具有较高的可配置型，降低了系统内部的耦合性、增加了系统的可维护性。通过该系统的开发与改进，可以得出以下结论。

(1)系统开发的目的是为了满足用户的需求，只有真正的理解了用户的系统建设目标，才能以用户的目标为出发点来确定系统中所需要的功能，所开发的系统才是有效的系统。

(2)基于 J2EE 技术的很多开源框架与 J2EE 的标准实现相比更适合于系统开发。通过有效的框架的搭建可以简化问题解决的繁杂程度，明晰系统的组织结构，同时还可以提高系统的可配置性、可扩展性与可重用性，有利于应用系统的灵活配置、扩展以及后续开发。另外，结构较好的框架还可以在相似的情况下重用，

提高开发的效率，增加开发人员的经验。

(3)设计模式都是很好的设计与开发经验的总结，将其有效而适当地应用于系统的实际开发中，能够有效的改善系统结构及代码结构，提高系统的可重用性及扩展性。

(4)有效的数据库设计可以保证系统的稳定性，数据库是系统正确有效运行的保证，在系统设计过程中在对系统实际需求的基础上对数据库进行认真的设计。

我国地域辽阔、地形复杂、人口散居，特别是偏远山区，利用地面无线电视广播或有线电视广播是难以覆盖的。而本系统是通过卫星广播的，卫星广播不受高山、森林等地理障碍的影响，扩大了紧急预警信息的公众覆盖面，也为一切预警信息发布提供了有效的发布渠道，是现阶段行之有效的预警信息传播手段。该系统的应用，大大缩短了灾害预警信息的传播时间，为突发紧急公共事件赢得宝贵的应急响应时间，是建设和谐社会的具体体现。

6.2. 展望

以上是一些经验的总结与开发系统过程中研究与解决相关问题的心得。但是现在的 CMMB 紧急广播信息发布系统还不够完善，还有很多可以加以改进的地方。目前系统用户较少，只有几个省市在做试点，而且很少存在并发访问的问题，如果系统的用户不仅仅局限在试点办公室，多用户访问时的性能问题在以后的系统建设过程中是需要考虑的。随着 CMMB 紧急广播信息发布系统的长时间运行，数据库中会存在海量数据，如何应对海量数据的存取是需要考虑的。部分功能模块也可以更加灵活一些。希望在以后的开发过程中可以对这些问题进行很好的解决。

基于 CMMB 的紧急通知发布系统是一个系统工程，需要不断的完善和不断的探索利用新技术和新方法改进预警信息发布效果。本论文工作至少在以下几个方面存在较大的不足：

(1)信息发布终端接收设备的性能有待提高，时常会有信息不能及时收到，信息安全性尚需进一步提高。

(2)该系统距离完全业务化使用还有一段距离，需要在统一发送编码，规范发布格式，统一发送指令等方面做更加深入的工作。

基于以上这些考虑，下一步主要工作为进一步完善系统的业务功能，使之能够顺利在国内推广，为紧急信息以及其他灾害信息的发布的传播时间大大缩短，最大程度地降低灾害给国家和人民带来的财产损失。

7. 参考文献

[1]. 张胜波，潘志文．CMMB 体系架构及其核心技术 STIM I． 信息化研究．2009 第 35 卷第 8 期

[2]. 韩建华．CMMB 系统回传通道方案的探讨及实现．中国有线电视．1007-7022(2010)01-0028-03

[3]. 秦大河，孙鸿烈，孙枢等．中国气象事业战略研究(总论卷)[M]．北京：气象出版社。2004：12-21．

[4]. 张迪．气象灾害预警信息发布系统若干关键技术研究．2009

[5]. 王峰 气象预警信息发布系统的设计与实现． 2008

[6]. 马翠平，杨荣芳，刘建文等．气象．2008-09

[7]. 石永怡．中国气象频道与国外气象频道的对比分析．媒介．2009-09

[8]. 赵翮， 吴力夫， 苗勃等．美日紧急广播系统浅析与思考．国家广播电影电视总局广播科学研究院． 2010-03

[9]. 刘文军, 李靖等．泰州市气象灾害预警与信息发布系统．中国气象学会．2008 年

[10]. 国家广播电影电视总局．中华人民共和国广播电影电视行业标准 2006-10

[11]. 阎宏．Java 与模式．北京；电子工业出版牡．2002：427-545

[12]. 王俊标．精通 Java Web 开发一基于 Struts EJB Hibernate Jbuilder．北京：人民邮电出版社，2007：91-92

[13]. Trygve M. H. Reenskaug．MVC—XEROX PARC 1978-79

[14]. Apache 公司．Struts2.0.11．http://struts.apache.org/ 2007-02

[15]. 李刚．Struts2 权威指南．北京：电子工业出版社．2009

[16]. 廖雪峰．Spring2.0 核心技术与最佳实践．北京：电子工业出版社 2007 年：4-6

[17]. 罗时飞．精通 Spring．北京：电子工业出版社 2005：73-75

[18]. springsource SPRING2.5． http://www.springsource.org/ 2008-10

[19]. Robert Cecil Martin. Designing Object Oriented C++ Applications Using The Booch Method，Prentice Hal l 1995 347-351

[20]. Rob Barrop．Jan Machacek．Pro Spring．Apress Publishing Inc．2005：Chapter 4

[21]. 刘天阳,陈锋. 一种基于 AOP 的软件演化策略设计与实现. 大连理工大学学报 2007 第 47 卷第 2 期：270-272

[22]. 庄巧莉，吴国华，冉宇瑶. 面向 AOP 构建可扩展的系统. 计算机应用与软件. 2007，第 24 卷第 8 期：78-82

[23]. Christian Bauer，Gavin King. Hibernate in Action. Manning Publications，2005：Chapter 1. 41

[24]. 沈洋. ORM 技术在机场信息系统中的应用. 计算机工程. 2007 年，第 33 卷第 19 期：241

[25]. 杨飞，袁建华. Hibernate 框架在权限管理系统中的应用. 微电子与计算机. 2007 年. 第 24 卷第 2 期：207

[26]. 卢卫杰. 基于轻量级 J2EE 框架的 WEB 信息系统的研究和应用 2007-12

[27]. 史周军，升晓俊. 基于元数据的对象关系映射研究. 计算机科学. 2005，V01. 32 NO. 5：95-97

[28]. 陈天河. Hibernate 项目开发宝典. 北京：电子工业出版社 2007

[29]. 陶勇，李晓军. Hibernate ORM 最佳实跻. 北京：清华大学出版社. 2007： 112-137

[30]. 付京周. 精通 Hibernate-Java 数据库持久层开发实践. 北京：人民邮电出版社，2007：163-169

[31]. JBoss. Hibernate3.2. http://www.hibernate.org/. 2006-10

[32]. C. T. Arrington，Syed H. Rayhan. Enterprise Java with UML。Second Edition [M]，John Wiley&Sons，2003，325-356，420-465

[33]. Alistair Coekburn. Writing Effective Use Cases[M]，Addison-Wesley，2001，15-21

[34]. 王佳冠. 基于 J2EE 的短信服务平台的设计及实现. 2006-12

[35]. Martin Fowler,王怀民，周斌. 企业应用架构模式[M]，机械工业出版社，2004，12—13

[36]. 中华人民共和国广播电影电视行业标准 移动多媒体广播 第一部分：广播信息帧结构、信道编码和调制 2006

[37]. 中华人民共和国广播电影电视行业标准 移动多媒体广播 第二部分：复用 2006

[38]. 中华人民共和国广播电影电视行业标准 移动多媒体广播 第三部分：电子业务指南 2006

[39]. 中华人民共和国广播电影电视行业标准 移动多媒体广播 第四部

分：紧急广播 2006

[40]. 孙鑫. Java Web 开发详解—xml+Servlet+JSP 深入剖析与实例应用. 电子工业出版社. 2009-06

[41]. 张龙样编著. UML 与系统分析与设计. 人民邮电出版社. 2002-09

8.致谢

在完成本论文的过程中，得到了江老师的悉心指导和大力支持。从理论准备和课题思考阶段开始，江老师就给予了我悉心的指导;在论文的撰写阶段，江老师从繁忙的工作中抽出时间仔细地审阅文章，并提出了很多宝贵而中肯的意见。江老师严谨的治学态度、认真的工作作风深深影响了我，为我以后的科研工作打下了良好的基础。在此，向江老师表示衷心的感谢！

9.个人简介

个人简历：

罗淦坤，男，安徽黄山人，1986 年 9 月出生。于 2007 年 6 月在安徽农业大学获得工学学士学位，于 2007 年 9 月进入安徽农业大学研究生学院攻读工学硕士学位。研究方向是农业机械化工程计算机应用工程。就读期间在深圳市世纪恒诺科技有限公司担任 java 工程师职务。

研究成果：

(1)参加国家广播电影电视总局 CMMB 紧急广播信息发布系统的研发

(2)负责中国气象局应急服务信息系统的研发

10. 在学期间发表的论著及科研成果清单

罗淦坤，江家伍.CMMB 紧急广播信息发布平台设计与实现.计算机技术与发展，编号：#1003354