Ni wanba2 分类号：_____ 单位代码：_____

密　级：_____ 学　　号：_____

浙江大学

硕士学位论文

中文论文题目 ：手机多媒体素材管理系统的设计与实现

英文论文题目：_____

申请人姓名：_____

指导教师：_____

合作导师：_____

专业学位类别：工程硕士

专业学位领域：软件工程

所在学院：软件学院

论文提交日期　　年　　月　　日

题目

作者姓名　浙江大学

手机多媒体素材管理系统的设计与实现

论文作者签名:_____

指导教师签名:_____

论文评阅人 1：_____

评阅人 2：_____

评阅人 3：_____

评阅人 4：_____

评阅人 5：_____

答辩委员会主席：_____

委员 1：_____

委员 2：_____

委员 3：_____

委员 4：_____

委员 5：_____

答辩日期：_____

Author's signature:_____

Supervisor's signature:_____

Thesis reviewer 1：_____

Thesis reviewer 2：_____

Thesis reviewer 3：_____

Thesis reviewer 4：_____

Thesis reviewer 5：_____

Chair：_____
(Committee of oral defence)

Committeeman 1：_____

Committeeman 2：_____

Committeeman 3：_____

Committeeman 4：_____

Committeeman 5：_____

Date of oral defence：_____

浙江大学研究生学位论文独创性声明

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。除了文中特别加以标注和致谢的地方外，论文中不包含其它人已经发表或撰写过的研究成果，也不包含为获得 浙江大学 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。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 浙江大学 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 浙江大学 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和传播，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适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘要

近年来，随着手机功能的日益强大和 3G 时代的到来，我们可以随时随地拍摄一些新闻素材及时报料给电视台不再是梦想，传统电视台主要通过记者和摄像人员在外采集新闻，这种方法由于设备的不便利性和人力资源的局限性，所搜集到的新闻素材也是相当有限，所以电视台在此基础上推出了“人人都是记者“的的概念，想通过一些普通用户把及时遇到的有价值的新闻拍摄下来，上传给电视台用于新闻播出，当前手机终端应用主要有短信、彩信和 WAP 等，因此上传回来的素材具有多样性，并且电视台也需要进行过滤筛选，所以在此基础上需要建立专门为其服务的管理系统，同时此款系统也打算支持前台 WAP 手机网站。

手机多媒体系统是基于.NET 的应用系统开发过程及其相关技术，结合实际的研发项目需求和业务流程，运用 ASP.NET Web 技术，SQL Server 2008，AJAX 等技术，采用了 B/S 开发模式和 WebService 服务技术，并且系统架构灵活运用了设计模式，运用了比较流行和时尚的设计思想对原有的传统.NET 三层架构进行了新的分层，优化了数据库访问层，操作层和业务逻辑层，并采用了面向接口编程思想。这种方法极大降低了层与层之间的耦合度，有利于数据库的迁移与整合。

本论文重点是论述基于.NET 体系下的手机多媒体素材管理系的设计与实现，首先介绍对系统开发所用到的关键技术，其次按照系统开发的思想，进行需求分析，架构设计，系统实现等，最后本系统前期主要分为频道管理，报料系统，用户管理和系统配置，由于采用设计模式的特殊性，我们后期还可以在此系统上增加新的平行系统，并不改变原有逻辑结构和数据结构。目前该系统的雏形已经完成。系统运行状况良好，表明总体系统方案设计合理可行。

关键词：手机多媒体，.NET，设计模式 面向接口编程

Abstract

In recent years, with mobile phones become more powerful and the 3G era, we can shoot anywhere in time Baoliao some news material to television is no longer a dream, mainly through the traditional television reporters and camera crews out collecting information, this approach because equipment and human resources, not the convenience of the limitations of the information collected material is quite limited, so television launched on this basis "everyone's" in concept, would like to in a timely manner through a number of regular users experience valuable information recorded by the upload to broadcast television for news, current mobile terminal application mainly SMS, MMS and WAP, etc., so upload the material to come back with diversity and television stations also need to filter screening On this basis, therefore need to establish specialized services for the management system, while Cikuan system also intend to support the front WAP phone site.

Mobile multimedia systems is based on. NET applications and related technology development process, combined with the actual needs of R & D projects and business processes, using ASP.NET Web technologies, SQL Server 2008, AJAX and other technologies, using B / S development model and WebService service technology, and system architecture flexibility in the use of design patterns, use of a relatively popular and stylish design of the original tradition. NET for a new three-tier hierarchical, optimizing the database access layer, business logic layer and the operating layer, and adopted for the interface programming ideas. This approach greatly reduces the coupling between layers degree, is conducive to the database migration and integration.

This paper focuses on paper-based. NET system for mobile multimedia material under the management of the Department of Design and Implementation, first introduced by the system used in key technology development, followed by system development in accordance with the idea of a needs analysis, architecture design, system implementation, etc. This system is divided into pre-channel management, Baoliao system, user management and system configuration, as with design patterns are unique, we can later on this system to add a new parallel system does not change the original structure and logic data structure. At present the system prototype has been completed. System running in good condition, indicate overall system design is reasonable and feasible.

Key Words : Mobile multimedia,. NET, Design Patterns for Interface Programming

目录

图目录

表目录

第1章 绪论

1.1 课题背景

近几年来，无线移动通讯领域取得了极为快速的发展，移动通信技术同互联网技术出现了日益融合的趋势。这种发展在业务领域体现在手机由传统的语音通话工具逐步转移为用户的便携式智能平台，除通话之外的其它许多增值业务被手机支持及被用户广泛使用，如移动上网功能、移动收发邮件及移动游戏平台等[1]。伴随着3G时代的到来，手机多媒体将在我们生活中扮演着越来越重要的角色，而目前各电视台进行新闻报料主要还是录播的方式，滞后性很大，因为新闻具有不可预见性，所以要想进行实时播出与报导的难度比较大，而利用手机多媒体上传拍照、拍视频进行实时上传，具有便利性，实时性，角色的大众性以及成本的优越性，但上传的素材根据路径和格式的不同，以及数据量的庞大，这给管理上带来极大的不便。

基于此，如何将上传得到的素材进行分类管理、推送、播出，将是一个比较有具体实际意义的项目，也将具有广阔的应用前景。手机多媒体素材管理系统是针对杭州电视台想要推出的新闻报料栏目而发开的，主要是帮助电视台栏目及时管理从大众手机端获取的多媒体素材（文字，图片，视频等），并进行详细的分类管理，同时也支持并管理即将开发的前台WAP手机网站。

1.2 课题的意义及初研究分析

1.2.1 手机多媒体素材后台管理系统项目意义

手机多媒体素材后台管理系统的本质是以办公智能化，提高办公效率，办公决策能力为目的。本项目通过与电视台新闻报料栏目实际相结合中，充分展示了其办公效率性，决策性以及智能性，传统媒体在做播出时候需要专门的记者等采访人员出去采访，而此款系统是为手机多媒体上传服务的，电视台通过做宣传推出各种奖励来吸引来自民间大众的新闻，趣事等，从而利用手机这种大众性的工具来为电视台服务。

由于这种在民间大众的可行性与方便性，所以获取的新闻资源将非常丰富，种类也将非常繁多，届时电视台在电视频道新闻节目上对其报导需要进行详细的

处理分类和审核等，如果不借助于外部工具，可以说此类新闻播出节目将无法进行，基于此我们电视台需要开发一个强大的后台系统对其进行支持，将其所收到的报料内容在后台充分展示也可以快速将其分类，并可以进行审核，推荐，锁定等操作，此后台项目还为电视台即将开发的前台互联网报料网站留好界面，预留滚动，头条，热点，关注，精彩等操作，可以说此款系统将来可以应用到有此类报料节目的各大电视台，新媒体当中。

1.2.2 手机多媒体素材后台管理系统研究现状

据本人调查研究，目前国内各大电视台还是广泛使用的传统摄像录拍技术，一些突发事件的拍摄也是采用此技术，相对来说比较具有滞后性，所以利用手机多媒体获取这些突发事件在杭州电视台是最具创新理念，也具有很大的探索性与挑战性。

1.3 本论文的主要研究内容和章节安排

手机多媒体素材管理系统将采用.Net 框架体系结构，众所周知，传统的.Net 框架下三层架构比较成熟，即数据访问层，业务逻辑层和用户表示层，但由于杭州电视台此项目的特殊性，因为电视台要考虑到后期与其它系统的对接与整合，包括数据库类型的不同等，所以这里结合实际情况，我们将数据访问层与业务逻辑层进行了特别修改，打造成为适合此项目的最佳系统架构。同时我们也在表示层加入一些时尚的元素，运用时下一些比较流行的技术如 AJAX，JavaScript 等，增强用户体验，使此后台项目更具有可操作性。

第 2 章 与本课题相关的技术

2.1 .NET 架构技术

手机多媒体素材管理系统是完全基于.net 平台和采用.net 框架而开发的，微软.NET 框架是用来构建、配置、运行 Web 服务及应用程序的开发平台。.NET 框架中采用了不少全新的技术，带来了许多决定性的、较深层次的创新[1]。. NET 框架的体系结构包括六部分：程序设计语言及公共语言规范(Common LanguageSpecification，CLS)、应用程序平台(ASENET 和 W'mdo 哪应用程序等)、ADO．NET 和 XML、基础类库、公共语言运行库(Common Language Runtime，CLR)、程序开发环境(Visual Studio)。

.NET 框架的关键组件是公共语言运行库和.NET 框架类库(包括 ADO.NET、ASP．NET、Windows 窗体)。. NET 框架提供了托管执行环境、简化的开发和部署，以及与各种语言的集成[2]。

(1)公共语言运行库

．NET 框架提供一个称为公共语言运行库的运行环境，它运行代码并提供使开发过程更轻松地服务。公共语言运行库负责运行时的服务，比如语言集成、安全增强，以及内存、子线程和线程的管理。另外，它在开发时也能发挥作用。在这个时候，具有诸如有效管理、强类型命名、跨语言异常处理、动态绑定等功能。

公共语言运行库的两个主要目标：

1．提高应用程序运行的稳定性和安全性。

2．减少应用程序开发者所必须编写的冗长而又容易出错的底层代码的容量。

(2)．NET 框架类库

基本类提供标准的功能，输入／输出、字符串操作、安全管理、线程管理、文本管理、用户界面设计的底层功能[3]。其他所有类库都建立在这个基本类库之上。ADO．NET(ActiveX Data Objects．NET)[4]数据访问类支持持续的数据管理。

它还包括 SQL 类，可以通过标准的 SQL 接口进行持续的数据存储操作。

XML 类可以实现 XML 数据操作和 XML 搜索和转换。

ASP.NET 类可以支持 Web Forms 应用程序和 Web Service 的开发。

Windows Forms 类支持基于 Windows 的智能客户端应用程序的开发。

所有这些类库结合在一起，提供一个跨所有．NET 框架所支持语言的公共、

一致的开发接口，并且采取清晰而有条理的方式对类库进行分组和描述，这样开发者能很容易的找到他们的应用程序所需要的大多数功能。

事实上，ASP．NET 是真正从底层被创建的，它是一种彻底不同的代码，是在 CLR 基础上和 XML 基础上，以及在所有其他．NET 技术基础上构建的。ASP.NET
是.NET 开发平台的一个部件,用来开发驻留在微软 IIS(Internet Information Server)上并使用诸如 HTTP 和 SAOP(简单对象访问协议)等 Intemet 协议的 Web 应用程序，这种应用程序有两种基本类型，即 Web Forms 应用程序和 Web Service。

(3)ADO．NET

ADO．NET(ActiveX Data Objects．NET)是微软在．NET 平台下提出的访问数据库的模型，ADO．NET 技术支持结合松散的数据访问需求、多层 Web 应用程序及 Web 服务，它提供了真正意义上的独立于任何数据源的数据访问。因此它可以用于多种不同的数据源，用于 XML 数据，用于管理应用程序本地的数据。ADO．NET 技术的出现，统一了昔日混乱的数据访问技术标准，提供了一个统一的访问接口。ADO．NET 结构包括两个核心组件：DataSet 和．NET 框架数据提供程序，后者是一组包括 Connection、Command、DataReader 和 DataAdaper 对象在内的组件。设计 ADO．NET 组件的目的是为了从数据操作中分解出数据访问。

DataSet 是 ADO.NET 的断开式结构的核心组件。DataSct 的设计目的很明确：为了实现独立于任何数据源的数据访问。因此，它可以用于多种不同的数据源，用于 XML 数据，或用于管理应用程序本地的数据。DataSet 与数据源是断开连接的，可将 DataSet 视为从数据库检索出的数据在内存中的缓存。DataSet 包含一个或多个 DataTable 对象的集合，这些对象由数据行、数据列及主键、外键、约束和有关 DataTable 对象中数据的关系信息组成。ADO．NET 结构的另一个核心组件是．NET Framework 数据提供程序，其设计目的相当明确：为了实现数据操作和对数据的快递、只读访问。

．NET 框架数据提供程序的核心元素是 Connection、Command、DataReader 和 DataAdaper 对象[5]。其中 Connection 对象提供与数据源的连接。Command 对象能够访问用于返回数据，修改数据，运行存储过程，以及发送或检索参数信息的数据库命令。DataReader 对象从数据源中提供高性能的数据流。DataAdaper 对象提供连接 DataSet 和数据源的桥梁。DataAdaper 对象使用 Command 对象在数据源中执行 SQL 命令，以便将数据加载到 DataSet 中，并使对 DataSet 中数据的更

改与数据源保持一致。

在.NET 平台中，.NET 框架占据着核心的位置，它是整个.NET 平台的关键支撑，是为众多高级语言(如 C#, Managed C++,Visual Basic .NET 等)和应用程序模型(如 Windows 窗体、ASP.NET、Web 窗体、XML Web 服务等)提供各种服务的重要基石。脱离.NET 框架来谈.NET 平台，难免不陷入空中楼阁的尴尬境地。实际上，在.NET 平台中，任何一门编程语言提供的功能都只是.NET 框架下一个子集的映射，具体的语言已经退居为一个语法表达的层次了。除了编程语言外，各种类型的.NET 应用程序在设计、开发、测试、部署、运行等诸多环节也和.NET 框架密切相关。如果没有对.NET 框架的深刻把握，学习再多的.NET 应用程序模型“开发技巧”都将只是徒劳—皮之不存，毛将焉附?因此，不管是学习 Windowns 窗体、还是 ASP.NET Web 窗体、抑或是 XML Web 服务，都必须从.NET 框架开始迈出坚实的一步[6]，在这个环境中，各个相关设备、网络、连接到网络的计算机互联在一起相互协作，进行网络计算[7]。

2.2 AJAX 技术

AJAX 是 Asynchronous JavaScript+XML 的简写，它是 Adaptive Path 公司 Jesse Garrett 在其发表的文章 AJAX：A New Approach to Web Applications 中所定义的[8]，它的意思是异步 JavaScript 和 XML 技术。AJAX 代表 Web 编程理念的一种转变，不是一个技术，它实际上是几种技术的结合体，每种技术都有其独特之处，合在一起就成了一个功能强大的新技术[9]。AJAX 由以下几种技术构成：

(1)基于 XHTML 和 CSS 标准的表示；

(2)使用文档对象模型(DOM)进行动态显示和交互；

(3)使用 XML 和 XSLT 进行数据交互和操作；

(4)使用 XmlHttpRequest 进行异步数据接收；

(5)使用 JavaScript 将上述技术绑定在一起。

AJAX 技术是通过在用户和服务器之间引入一个 AJAX 引擎，这个引擎负责绘制用户界面以及与服务器端通信。AJAX 引擎允许用异步的方式实现用户与服务器的交互而不用等待服务器的通信，打破了用户与服务器交互传统的“开始—停止—开始—停止”交互过程。提供与服务器异步通信的能力，从而使用户从“开始—停止”的循环中解脱出来。

最有名的 Ajax 应用几乎都是 Google 公司的大作，包括 GMail，Google

Suggest，Google Maps，可以说这是最广为人知的几个 Ajax 应用。2004 年初，Google 就推出了 beta 版本的 GMail 服务。除了大容量以外，GMail 最出色的地方是它的用户界面。它允许用户一次打开多个邮件，并且即使用户在编写邮件时，邮件列表也能够自动更新。与其他的邮件系统相比，这是一个巨大的进步。与其他 Web 界面模仿桌面邮件客户端的邮件服务相比，GMail 没有依赖重量级的、容易出问题的 ActiveX 控件和 Java Applet，但在功能上却毫不逊色。这样做所带来的好处就是完全的跨平台，可以在任何平台、任何地点使用 GMail 的服务。

此后，在提供更加丰富的交互性方面，Google 走得更远。例如，当用户键入字符时，Google Suggest 可以为用户提供与输入字符相符的提示，帮助他们完成想要键入的搜索字符串；Google Maps 可以执行交互式的、可缩放的基于位置的搜索。

以上这些应用大都还只能算是初步的尝试。它们仍然是瞬态应用，但是 Google Spreadsheets 在线电子表格和被 Google 收购不久的 Writely(现在，这两个应用已经被整合成了 GoogteDocs&Spreadsheets)，初步展示了 Ajax 技术构建独占应用的能力。

2.3 设计模式的运用

2.3.1 为什么要用设计模式

当今，软件开发已经有了巨大的变化。一个主要的变化就是结构化编程的介绍，结构化编程把应用程序分成很多功能，每一个功能执行一个特定的任务，每个功能都要因它将要执行的任而被命名，紧接着就是用面向对象的方式编程。现在开发者都依赖类来写他们的代码，用这些从现实中模型化过的类来编程。

而这些方法的主要目标都是帮助在团队内部或跨团队的沟通。“一个客户是怎么开户的？”使得一个开发者和另一个开发者和终端用户沟通起来非常容易。然后用类来模型化。而设计模式所要讲的就是沟通，它能帮助我们开发者更好的明白和更清晰的描述一段被写出的代码。模式不是描述代码，它是可替代的并允许开发者之间通过沟通问题的机制来使得问题得以解决。

设想有两个木匠，他们在讨论如何连接去使用一组抽屉，一个人问：“ 你准备用什么顺序连接这些抽屉”另一个答道 “我要拿两块木头，砍掉大约两厘米，在四十五厘米深度打角，然后再在四十五深度砍掉四厘米，然后上下四十厘米处，然后.....”

第二个人要描述的是如何去燕尾接合，在木工中是一个标准的接合方式，这样的描述不是目标沟通，实际上他隐藏了沟通，如果第二个人说“我要使用一个燕尾接合”那就会传递更多信息用很少的话。

在第二个例子中，木工正在描述一个正常的连接方式，并且基于他们的理解做出的描述。 例如每一个复杂的结合，结合的力度等等。这就是软件开发中设计模式应用的情况，如果一个开发者说“我在使用一个策略模式”或“我在使用一个工厂模式”传递给另外一个理解这个模式的并且知道如何使用的开发者[10]。

之所以使用设计模式的目的是为了适应未来的变化，变化之所以存在是因为一切的事物都具有不可预见性，如果具有可预见性，则不能称其为变化。为什么要使用设计模式，本人在实践以及此项目中有如下体会：设计模式是为了是设计适应变化；设计模式是重构的工具；设计一开始就要保持流畅、简单，并且具有可持续性；最后不能过度使用设计模式。

2.3.2 设计模式的理解和选择

“模式不是解决方案，而是在某种环境中权衡各方面利弊的一种方案的选择，这种选择是这些利弊平衡的结果，获得好处的同时需要付出代价，并且结果中有有利的方面，也有不利的方面。”当然有人也认为模式是解决方案，例如，在《在软件开发中理解和使用模式》一文中，Dirk Riehle 和 Heinz Zullighoven 给出了这样一个定义;模式是从解决具体问题抽象出来的，这种具体问题在特定的上下文中重复出现。也就是说，每个具体形式都对一种重复的问题采用重复的解决方案。然而，模式不仅仅是解决方案，正是因为很多人认为设计模式仅仅是解决方案，所以才导致设计模式的滋用和设计过度。这里我们根据“Patterns and Software: Essential Concepts and Terminology”一文来讨论模式的定义。

在 GOF 的设计模式描述中，没有显示的场景，只有“动机”[11]· 其中包含了场景的基本内容，但通常是采用一个具体的实例进行说明，而在“适用性”中总结并给出一般性的适用范围。需要强调的是，在使用设计模式之前，一定要分析问题所处的实际场景，查看是否符合潜在选择的模式存在的语境。如果不适合，可能选择的模式有问题。有时，尽管模式的解决方案可以解决问题，但这个方案很可能不是一个很好的选择，可能增加系统的复杂程度。

例如，创建型模式的场景是需要创建实例的类，这些类之间的关系和使用这些类的客户。图 2.1 所示为一个实际的场景。

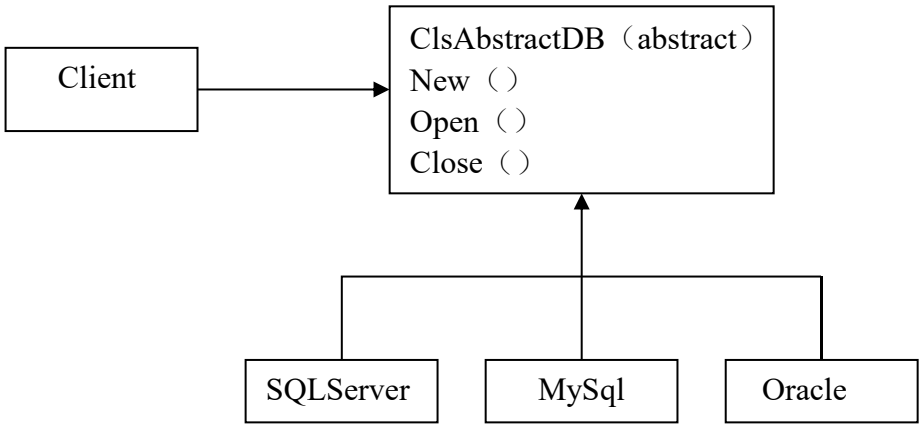


图 2.1 场景图

很显然，clsAbstractDB 是一个抽象类，而客户只知道它。这个场景说明 client 必须以某种方式获得 clsAbstractDB 子类的一个实例，这时我们需要采用某种创建型模式。需要注意的另一个问题是，如果实际的场景本身有设计缺陷，那么首先需要优化这些设计。还是上面的例子，这样的设计是可能的，如图 5.2 所示。

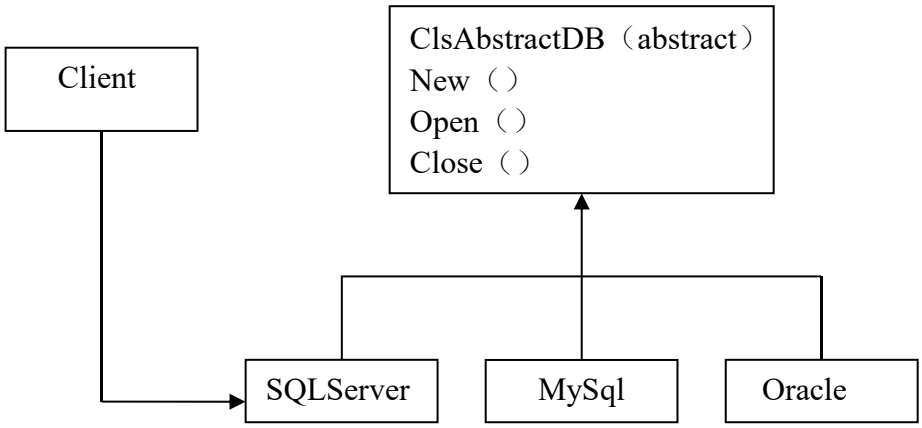


图 2.2 优化场景图

如果将这个模型作为场景，则没有应用设计模式的必要。因此，评价使用设计模式是否合适，还需要评价作为模式相关的场景设计是否合理。很多情况下，正是因为现有设计中存在缺陷和潜在的不足才需要引入设计模式。而引入设计模式势必要改变原有的结构，从而涉及重构的工作。在上面的例子中，设计的缺陷

是如果数据库换成 Oracle，则 Client 必须改变，而这正是我们需要避免的。

所以综上所述在本项目中，充分考虑到杭州电视台对此系统的可扩展性可维护性以及可变化性，故采用了抽象工厂模式，并结合实际的项目需求进行了调整与优化，做到灵活多变。

2.3.3 抽象工厂模式在本项目中的应用

抽象工厂的“抽象”来自于“抽象的产品角色”，而抽象工厂就是抽象产品角色的工厂。每一个设计模式都是针对一系列的问题的解决方案。抽象模式就是针对多个等级产品结构的系统而设计的。如图 2.3，

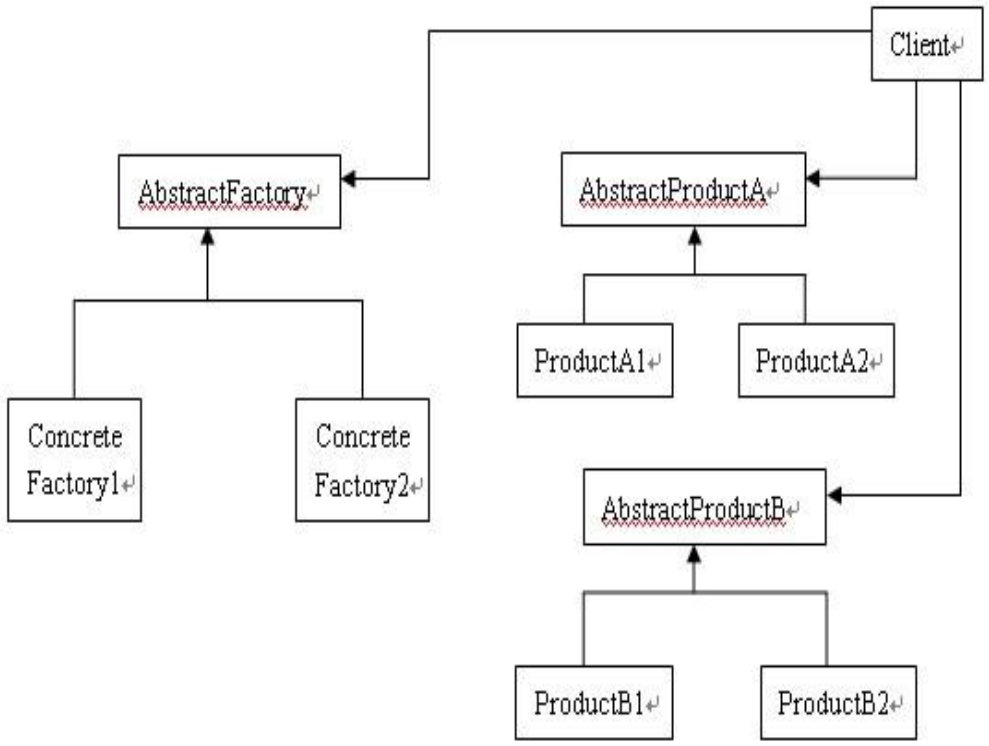


图 2.3 抽象工厂图

AbstractFactory:声明一个创建抽象产品对象的操作接口。

ConcreteFactory：实现创建具体产品对象的操作。

AbstractProduct：为一类产品对象声明一个接口。

ConcreteProduct：定义一个将被相应具体工厂创建的产品对象，以实现 AbstractProduct 接口。

Client：仅使用 AbstractFactory 和 AbstractProduct 类声明的接口

抽象工厂模式就是向客户端提供一个接口，使得客户端在不必知道产品的具体类型的情况下，创建多个产品族中的产品。

在本项目中，频道栏目是等级产品结构，而栏目下对应的众多子栏目以及栏目对应的众多子多媒体素材分类，更是等级产品结构，并且在开发用户角色管理时候也是采用权限的等级结构，所以该项目产品核心业务内容很符合抽象工厂模式的设计理念与思想。在本项目中抽象工厂分离了具体的类，使得本项目产品系列容易交换，只要更换相应的具体栏目工厂即可。其次也有利于产品的一致性，由于抽象工厂模式创建的产品必须符合相同的接口，任何子类中的特殊功能都不能提现在统一的接口中。所以在本项目中各工厂产品创建的如栏目或者多媒体素材都具有一致性，对于不同的产品的特殊功能，我们采用不同的接口，所以为了系统里特殊功能的实现，我们采用面向接口编程，加入了接口层。并且对数据库访问类进行优化，这样使得数据库操作类不再是一个抽象类，而是通过工厂获得相应的 IDbConnection、IDbCommand 和 IDataAdapter，具体的工厂作为参数传入。这样，数据库操作类与具体的数据库就没有任何耦合性。

2.3.4 模式的深入理解

模式理论的精髓之一就是模式的使用是有前提和代价的，模式是在某种前提下，综合各方面因素后考虑得出的结果。即在使用模式时总是要付出一定的代价，当然这种代价是可以接受的。如果某个模式在所有场合中的使用都是必然的，那么它就不能叫做模式了，而是一种必须遵守的法则。例如“面向接口，而非实现编程”，是法则而非模式。如前所述，模式是某种语境下的解决方案。在实际的项目中，是否采用模式取决于项目是否符合模式的语境，并且是否可以接受使用模式所带来的代价。很多情况下，模式的选择不是必然的。有时可以不采用模式也可以很好地解决问题;有时可以采用一种模式代替另一种模式，或者说在特定的语境下模式的使用不是惟一的。例如命令模式将行为抽象为类，模式的实现比较简单，也容易理解。那么这个模式是否可以在所有场合使用呢?显然不可。如果我们将所有类的方法都抽象为命令模式，这样程序是最灵活的。因为可以动态地增加行为，但代价是程序中各个类就不再具有业务含义。业务含义全部隐藏在命令对象中，这样做的结果是晦涩难懂，可维护性反而降低。

为了合理地使用模式，需要深入理解模式。模式不仅仅是解决方案，在使用模式时不能仅将注意力放在模式如何实现上，更重要的是理解模式存在的上下文

及相关的“作用力系统”。由于设计模式没有明确指出这两个部分内容，所以要真正领会在问题、动机和实现中隐含的相关思想。模式的使用既有好的效果，也会付出一定的代价。就设计模式来说，对象增加及对象关系复杂是得到好处的代价。

总之，设计模式不是万能的，不能指望设计模式的使用能提高开发的速度，或者项目的形象进度。当有很好的实践基础时，模式的使用就变成自然而然的事情[12]。

2.4 B/S 结构模式

近年来，随着 Intemet 技术的飞速发展以及网络的普及，基于 Web 的浏览器/服务器的结构得到了人们广泛的重视，基于浏览器/服务器结构的各种网站、应用程序正在越来越大的程度上取代传统 C/S 体系结构的软件，成为当前软件开发的主流。

Browser/Server(B/S)是一种由表示层(Browser)、功能层(Web Server)、数据库服务层(DB Server)构成的三层分布式体系结构，其数据及应用可通过不同平台、不同网络存取，与平台无关，伸缩性大[13]。B/S 结构模式使用基于超文本传输协议(HTTP，Hyper Text Transfer Protoc01)的消息传递机制。客户端通过浏览器访问服务器并发出服务请求，服务器进行相应的处理后将响应结果返回给客户端。与传统的 C/S 结构模式相比，B/S 结构模式具有以下特点：

(1)C/S 结构模式必须根据客户端所使用的操作系统及后台应用的不同，安装不同的软件；B/S 结构模式则极大的简化了客户端软件，只需安装 Web 浏览器作为执行客户端应用的平台。

(2)C/S 结构模式往往需要混合使用多种传输协议，而基于 B / S 结构模式的所有系统都使用 HTTP 协议进行通信。

(3)C/S 结构模式的系统维护和版本升级比较麻烦，而在 B / S 结构模式中，所有的维护和升级工作都是在 Web 服务器端进行的。用户访问网站时动态下载网页内容，保证了用户每次使用的都是最新版本。

2.5 本章小结

本章详细介绍了课题研究涉及的相关开发技术：首先介绍．NET 框架的核心组件是公共语言运行库和．NET 框架类库、接着介绍 AJAX 技术特点、以及设计

模式在本项目中的运用和对本项目的重要性，最后讲解了 B／S 结构的特征和本项目中运用 B/S 的优点。

第 3 章 手机多媒体素材管理系统架构分析与设计

3.1 系统需求分析

需求分析是软件生命周期中非常重要的一步，它最根本的任务是确定系统是否具有价值。从本质上说，需求分析要解决系统必须做什么的问题[14]。面向对象的系统分析和设计是应用系统开发的新途径，在管理信息系统开发中使用面向对象的系统分析和设计将加快系统开发速度，提高开发质量[15]。

由于考虑到电视台系统的多样性以及将来可能发生的变动，手机多媒体素材管理系统在充分了解到电视台各频道各栏目的需求后勇于创新，深入运用设计模式打造了一个灵活多变的系统架构，在此架构上我们后期可以无限扩展，不需要改变原来的业务逻辑代码与数据库，只需平行增加子系统的业务逻辑代码与数据库。这样我们可以大大提高电视台各部门的工作效率，降低电视台投入软件的成本，增长系统的使用周期。

3.2 系统架构设计

在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构。微软推荐的分层式结构一般分为三层，从下至上分别为:数据访问层、业务逻辑层(又或成为领域层)、表示层，如图 1.1 所示:

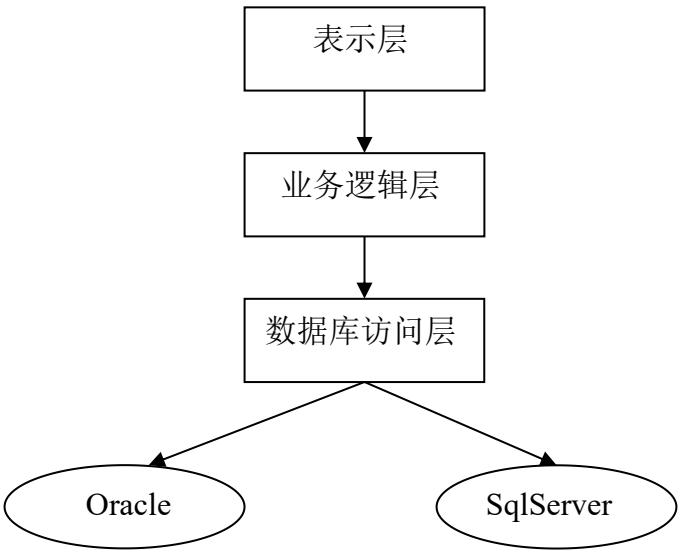


图 3.1 传统架构图

传统的.NET 三层架构中，业务逻辑层与数据库访问层职责相对比较混乱，一旦要求支持的数据库发生变化，或者需要修改数据库访问的逻辑，由于没有清晰的分层，会导致项目作大的修改。但是此项目由于是杭州电视台为其自己量身打造的独特的后台管理系统，并且需要与其它系统的对接，对前台互联网站的支持以及将来可能发生的数据库的迁移等。这就需要一个好的分层结构。

一个好的分层式结构，可以使得开发人员的分工更加明确。一旦定义好各层次之间的接口，负责不同逻辑设计的开发人员就可以分散关注，齐头并进。例如 UI 人员只需考虑用户接口的体验与操作，领域的设计人员可以仅关注业务逻辑的设计，而数据库设计人员也不必为繁琐的用户交互而头疼了。每个开发人员的任务得到了确认，开发进度就可以迅速的提高。

松散耦合的好处是显而易见的。如果一个系统没有分层，那么各自的逻辑都紧紧纠缠在一起，彼此间相互依赖，谁都是不可替换的。一旦发生改变，则牵一发而动全身，对项目的影响极为严重。降低层与层间的依赖性，既可以良好地保证未来的可扩展，在复用性上也是优势明显。每个功能模块一旦定义好统一的接口，就可以被各个模块所调用，而不用为相同的功能进行重复地开发。进行好的分层式结构设计，标准也是必不可少的。只有在一定程度的标准化基础上，这个系统才是可扩展的，可替换的。而层与层之间的通信也必然保证了接口的标准化。

所以本系统采用了多层结构，根据其业务需求，在传统三层架构上灵活运用，开发了适合此系统的最佳架构图，如下图 2.2：

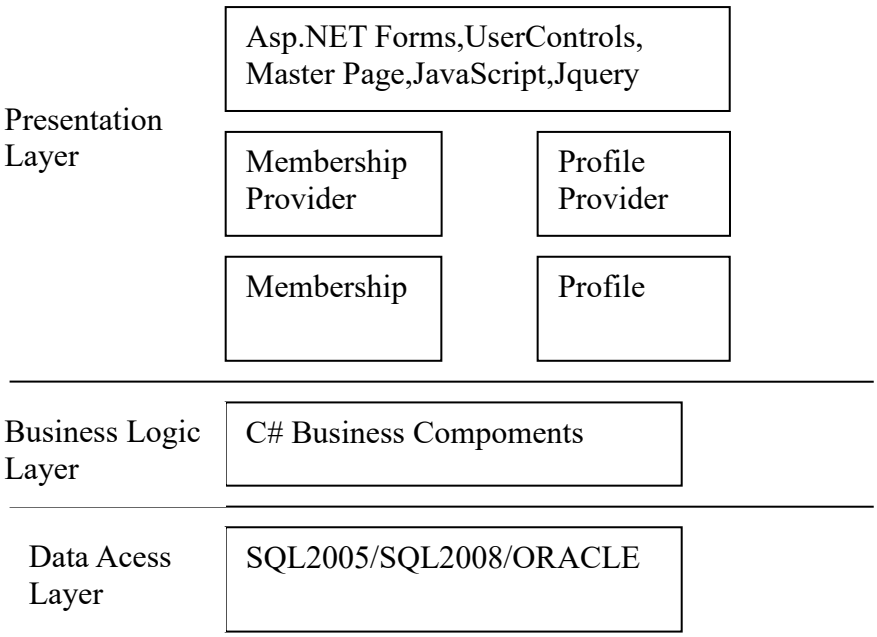


图 3.2 系统架构图

3.3 数据访问层设计

综合考虑其多种因素，我们在原有的三层架构基础上，对数据库层和业务逻辑分别做了改动，在数据访问层中我们采用了“面向接口编程”思想。如下图 2.3 所示：

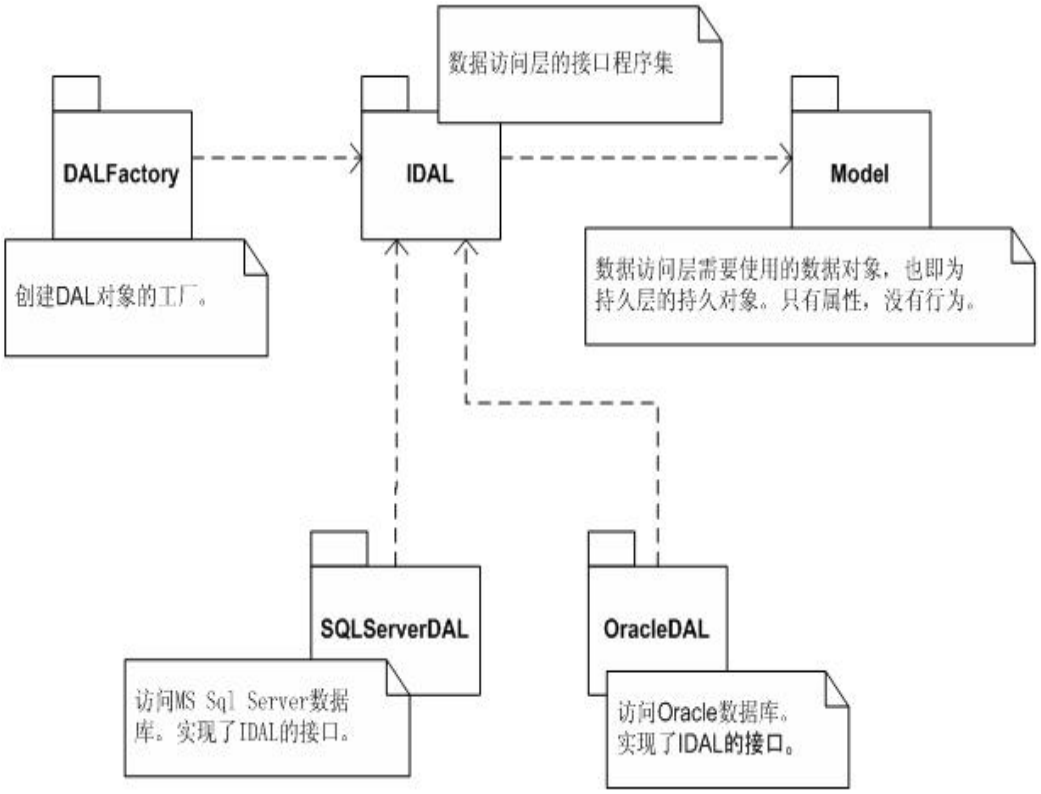


图 3.3 数据库访问层图

可以看到，在数据访问层中，抽象出来的 IDAL 模块，脱离了与具体数据库的依赖，从而使得整个数据访问层利于数据库迁移，并以 DALFactory 作为数据访问层对象的工厂模块，抽象出数据访问逻辑，专门管理数据操作层和 DAL 对象的创建，便于业务逻辑层的访问。无论是 SQLServerDAL 还是 OracleDAL 模块均实现 IDAL 模块的接口，其中包含的逻辑就是对数据库的 Select，Insert，Update 和 Delete 操作。因为数据库类型的不同，对数据库的操作也有所不同，代码也会因此有所区别。这里 DALFactory 管理接口模块主要是借鉴抽象工厂模式下，利用委托，反射和 Webconfig 配置文件来实现。

此外，在数据库层中采用“面向接口编程”有着以下优势：

(1)降低程序各部分之间的耦合性，使程序模块互换成为可能。这样客户无须知道自己使用的对象类型，只要对象有客户所期望的接口即可。并且客户也不需要知道对象是如何实现的，只要知道定义接口的抽象类。

(2)使软件各部分便于单元测试，通过编制与接口一致的模拟类(Mock)，可以很容易地实现软件各部分的单元测试。从而提高软件的可靠性，降低错误率。

(3)易于实现软件模块的互换，软件升级时可以只部署发生变化的部分，而不会影响其它部分。

3.3.1 数据访问层之数据库访问详细设计

从本部分开始，我将依次对各层进行代码级的分析，以求获得更加细致而深入的理解。在本项目中，由于引入了 ASP.Net 3.5 的一些新特色[16]，所以数据层的内容也更加的广泛和复杂，包括：数据库访问、Messaging, Membership, Profile 四部分。在这里中，将介绍有关数据库访问的设计。

在手机多媒体素材管理系统项目中，系统需要处理的数据库对象分为两类:一是数据实体，对应数据库中相应的数据表。它们没有行为，仅用于表现对象的数据。这些实体类都被放到 Model 程序集中，例如数据表对应的实体类，其类图如下图 2.4：

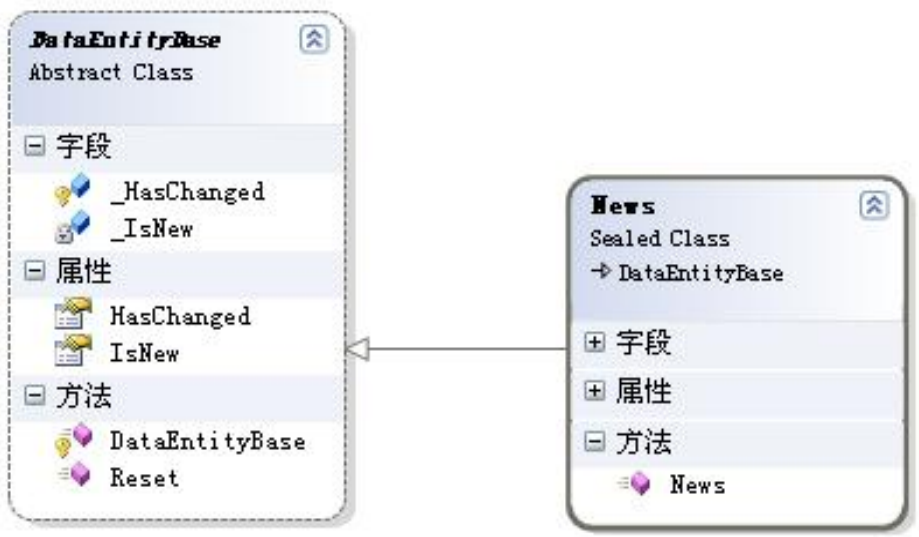


图 3.4 实体类图

这些对象并不具有持久化的功能，简单地说，它们是作为数据的载体，便于业务逻辑针对相应数据表进行读/写操作。虽然这些类的属性分别映像了数据表的列，而每一个对象实例也恰恰对应于数据表的每一行，但这些实体类却并不具备对应的数据库访问能力。

由于数据访问层和业务逻辑层都将对这些数据实体进行操作，因此程序集

Model 会被这两层的模块所引用。

第一类数据库对象则是数据的业务逻辑对象。这里所指的业务逻辑，并非业务逻辑层意义上的领域(domain)业务逻辑(从这个意义上，我更倾向于将业务逻辑层称为”领域逻辑层“)，一般意义上说，这些业务逻辑即为基本的数据库操作，包括 Select, Insert, Update 和 Delete 。由于这些业务逻辑对象，仅具有行为而与数据无关，因此它们均被抽象为一个单独的接口模块 IDAL，例如数据表 News 对应的接口 INewsDAL：

图 3.5 接口图

将数据实体与相关的数据库操作分离出来，符合而向对象的精神。首先，它体现了职责分离的原则。将数据实体与其行为分开，使得两者之间依赖减弱，当数据行为发生改变时，并不影响 Model 模块中的数据实体对象，避免了因一个类职责过多、过大，从而导致该类的引用者发生灾难性的影响。其次，它体现了”抽象”的精神，或者说是“面向接口编程”的最佳体现。抽象的接口模块 IDAL,与具体的数据库访问实现完全隔离。这种与实现无关的设计，保证了系统的可扩展性，同时也保证了数据库的可移植性。在本项目中，可以支持 SQL Server 和 Oracle,那么

它们具体的实现就分别放在两个不同的模块 SQLServerDAL, OracIeDAL 中。以 News 为例，虽然在两个模块中有不同的实现，但是他们又同时实现了 INewsDAL 接口，如下图：

图 3.6 接口操作图

从数据库的实现来看，本项目体现出了没有 ORM 框架的臃肿与丑陋[17]。由于要对数据表进行 Insert 和 Select 操作，以 SQL Server 为例，就使用了 SqICommand, SqIParameter,SqIDataReader 等对象，以完成这些操作。尤其复杂的是 Parameter 的传递，在此项目中，使用了大量的字符串常量来保存参数的名称。如下所示：

```
//   {0} TableName, {1} WhereClip
protected string Sql_Count = "SELECT COUNT(*) FROM {0} {1}";
//   {0} TopClip, {1} SchemaClip, {2} TableName, {3} WhereClip, {4} OrderByClip
protected string Sql_Select = "SELECT {0} {1} FROM {2} {3} {4}";

//   PKName  和传入字段双字段排序{0} Schema, {1} Table, {2} PKName, {3}
PageSize, {4} PageIndex * PageSize, {5} SchemaClip, {6} WhereClip, {7} OderClip0,
{8} OderClip1,
protected string Sql_Paging = "SELECT {0} FROM {1} WHERE {2} IN ( SELECT
```

```
TOP {3} {2} FROM ( SELECT TOP {4} {5} FROM {1} {6} {7}) TI {8} ) {7} ";
//    Insert, Update
protected string Sql_Insert = "INSERT INTO {0} ( {1} ) Values ( {2} )";
protected string SQl_Update = "UPDATE {0} SET {1} {2}";
```

在没有 ORM 的情况下，使用 Helper 类是一个比较好的策略，利用它来完成数据库基本操作的封装，可以减少很多和数据库操作有关的代码，这体现了对象复用的原则。本项目将这些 Helper 类统一放到 DBUtility 模块中，不同数据库的 Helper 类暴露的方法基本相同，只除了一些特殊的要求，例如 Oracle 中处理 boot 类型的方式就和 SQL Server 不同，从而专门提供了 OraBit 和 OraBool 方法。此外，Helper 类中的方法均为 Static 方法，以利于调用。SqlHelper 的类图如下：

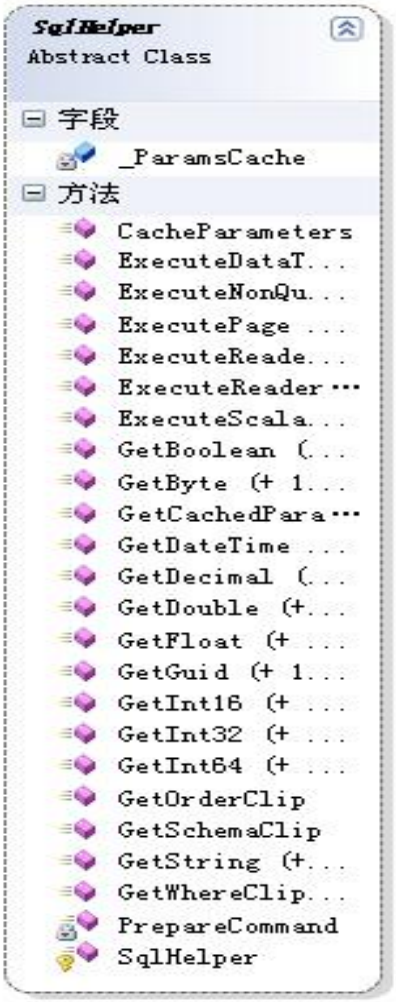


图 3.7 SqlHelper 类图

对于数据访问层来说，最头疼的是 SQL 语句的处理。在早期的 CS 结构中，

由于未采用三层式架构设计，数据访问层和业务逻辑层是紧密揉合在一起的，因此，SQL 语句遍布与系统的每一个角落。这给程序的维护带来极大的困难。此外，由于 Oracle 使用的是 PL-SQL，而 SQL Server 和 Sybase 等使用的是 T—SQL，两者虽然都遵循了标准 SQL 的语法，但在很多细节上仍有区别，如果将 SQL 语句大量的使用到程序中，无疑为可能的数据库移植也带来了困难。

现在我们已经有了数据实体，数据对象的抽象接口和实现，可以说有关数据库访问的主体就已经完成了。留待我们的还有两个问题需要解决：

1、数据对象创建的管理

2、利于数据库的移植

在本项目中，要创建的数据对象包括 News，NewsComment，SysChannel 等

在前面的设计中，这些对象已经被抽象为对应的接口，而其实现则根据数据库的不同而有所不同。也就是说，创建的对象有多种类别，而每种类别又有不同的实现，这是典型的抽象工厂模式的应用场景。而上面所述的两个问题，也都可以通过抽象工厂模式来解决。标准的抽象工厂模式类图如下：

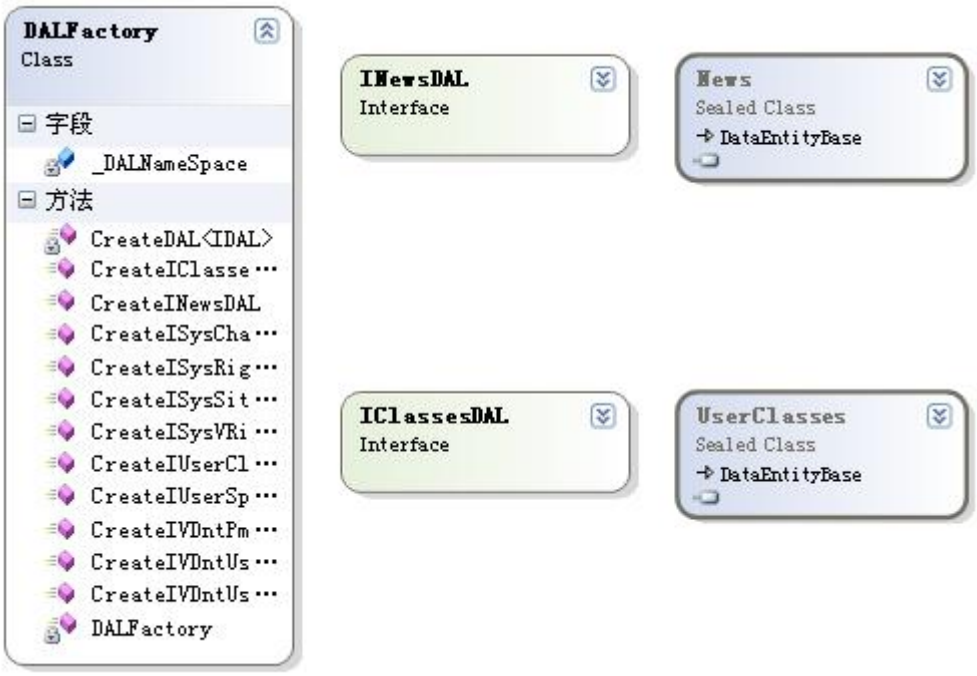


图 3.8 标准抽象工厂模式类图

例如，创建 SQL Server 的 Order 对象如下：

DALFactory factory=new SQLServerFactory();

News=factory.CreateNews();

要考虑到数据库的可移植性，则 factory 必须作为一个全局变量，并在主程序运行时被实例化。但这样的设计虽然已经达到了“封装变化”的目的，但在创建 DALFactory 对象时，仍不可避免的出现了具体的类 SQLServerFactory，也即是说，程序在这个层而上产生了与 SQLServerFactory 的强依赖。一旦整个系统要求支持 Oracle，那么还需要修改这行代码为.

修改代码的这种行为显然是不可接受的。解决的办法是”依赖注入。”依赖注入的功能通常是用专门的 IoC 容器提供的，在 Java 平台下，这样的容器包括 Spring, PicoContainer 等。而在.Net 平台下，最常见的则是 Spring.Net。不过，在本系统中，并不需要专门的容器来实现“依赖注入”，简单的做法还是利用配置文件和反射功能来实现。也就是说，我们可以在 web.config 文件中，配置好具体的 factory 对象的完整的类名。然而，当我们利用配置文件和反射功能时，
具体工厂的创建就显得有些“画蛇添足”了，我们完全可以在配置文件中，直接指向具体的数据库对象实现类，例如 Hcrt.SQLServerDAL.News。那么，抽象工厂模式中的相关工厂就可以简化为一个工厂类了，所以我将这种模式称之为“具有简单工厂特质”的抽象工厂模式，其类图如下：

图 3.9 简单抽象工厂类图

Hcrt.DAL.SqlServer 下的类完全取代了前面创建的工厂类体系，它是一个 sealed 类，其中创建各种数据对象的方法，均为静态方法。之所以能用这个类达到抽象工厂的目的，是因为配置文件和反射的运用，如下的代码片断所示：

```
public sealed class DataAccess
{
 private static readonly string _ConnString =
ConfigurationManager.ConnectionStrings[ConfigurationManager.AppSettings["default
Database"]].ConnectionString;
 private static read only string_DALNameSpace=
 ConfigurationManager.AppSettings["DALNameSpace"];
 private static IDAL CreateDAL<IDAL>(string dalName)
        {
            return
```

```
(IDAL)Assembly.Load(_DALNameSpace).CreateInstance(_DALNameSpace + "." +
dalName);
            }
```

在本项目中，这种依赖配置文件和反射创建对象的方式极其常见，包括 Hcrt.Common，DALFactory 等等。这些实现逻辑散布于整个系统中，也就是说，我们可以为整个系统提供类似于“Service Locator“的实现：

```
public static class ServiceLocator
{
private static readonly string dalPath=
ConfigurationManager.AppSettings["WebDAL"];
    private static readonly string orderPath=
ConfigurationManager.AppSettings["OrdersDAL"];
```

3.3.2 业务逻辑层设计

业务逻辑层(Business Logic Layer)无疑是系统架构中体现核心价值的部分。它的关注点主要集中在业务规则的制定、业务流程的实现等与业务需求有关的系统设计，也即是说它是与系统所应对的领域(Domain)逻辑有关，很多时候，我们也将业务逻辑层称为领域层。MartinFowler 在((Patterns of Enterprise Application Architecture》一书中，将整个架构分为三个主要的层:表示层、领域层和资料源层。作为领域驭动设计的先驭 Eric Evans，对业务逻辑层作了更细致地划分，细分为应用层与领域层，通过分层进一步将领域逻辑与领域逻辑的解决方案分离。

业务逻辑层在体系架构中的位置很关键，它处于资料访问层与表示层中间，起到了数据交换中承上启下的作用。由于层是一种弱耦合结构，层与层之间的依赖是向下的，底层对于上层而台一是“无知”的，改变上层的设计对于其调用的底层而言没有任何影响。如果在分层设计时，遵循了而向接口设计的思想，那么这种向下的依赖也应该是一种弱依赖关系。因而在不改变接口定义的前提下，理想的分层式架构，应该是一个支持可抽取、可替换的“抽屉”式架构。正因为如此，业务逻辑层的设计对于一个支持可扩展的架构尤为关键，因为它扮演了两个不同的角色。对于数据访问层而言，它是调用者；对于表示层而言，它却是被调用者。依赖与被依赖的关系都纠结在业务逻辑层上，如何实现依赖关系的解耦，则是除了实现业务逻辑之外留给设计师的任务。

3.3.2.1 业务逻辑层模式的应用

Martin Fowler 在《企业应用架构模式》一书中对领域层(即业务逻辑层)的架构模式作了整体概括，他将业务逻辑设计分为二种主要的模式:Transaction Script,Domain Model 和 Table Module[18]。

Transaction Script 模式将业务逻辑看作是一个个过程，是比较典型的而向过程开发模式。应用 Transaction Script 模式可以不需要数据访问层，而是利用 SQL 语句直接访问数据库。为了有效地管理 SQL 语句，可以将与数据库访问有关的行为放到一个专门的 Gateway 类中。应用 Transaction Script 模式不需要太多而向对象知识，简单直接的特性是该模式全部价值之所在。因而，在许多业务逻辑相对简单的项目中，应用 Transaction Script 模式较多。

Domain Model 模式是典型的而向对象设计思想的体现。它充分考虑了业务逻辑的复杂多变，引入了 Strategy 模式等设计模式思想，并通过建立领域对象以及抽象接口，实现模式的可扩展性，并利用而向对象思想与身俱来的特性，如继承、封装与多态，用于处理复杂多变的业务逻辑。唯一制约该模式应用的是对象与关系数据库的映像。我们可以引入 ORM 工具，或者利用 DataMapper 模式来完成关系向对象的映像。

与 Domain Model 模式相似的是 Table Module 模式，它同样具有而向对象设计的思想，唯一不同的是它获得的对象并非是单纯的领域对象，而是 DataSet 对象。如果为关系数据表与对象建立一个简单的映像关系，那么 Domain Model 模式就是为数据表中的每一条记录建立一个领域对象，而 Table Module 模式则是将整个数据表看作是一个完整的对象。虽然利用 DataSet 对象会丢失而向对象的基木特性，但它在为表示层提供数据源支持方而却有着得大独厚的优势。尤其是在.Net 平台下，ADO.NET 与 Web 控件都为 Table Module 模式提供了生长的肥沃土壤。

3.3.2.2 业务逻辑层详细设计

本系统架构设计中最核心的业务逻辑层，针对系统业务定义了相关的领域对象，但这些领域对象仅仅是完成了对数据访问层中数据对象的简单封装而已，其目的仅在于分离层次，以支持对各种数据库的扩展，同时将 SQL 语句排除在业务逻辑层外，避免了 SQL 语句的四处蔓延。

在上面我提到业务逻辑层中的领域对象仅仅是完成对数据对象的简单封装，

但这种分离层次的方法在架构设计中依然扮演了举足轻重的作用。以 News 类的 Add()方法为例，在方法内部引入了领域对象，并调用了 News 对象的 GetNews（）方法。如果没有在业务逻辑层封装对象，而是直接调用数据访问层的数据对象，为保证层次间的弱依赖关系，就需要调用工厂对象的工厂方法来创建 Hcrt.DAL.INewsDAL 接口类型对象。一旦数据访问层的对象被多次调用，就会造成重复代码，既不离于程序的修改与扩展，也导致程序结构生长为臃肿的态势。此外，领域对象对数据访问层数据对象的封装，也有利于表示层对业务逻辑层的调用。在三层式架构中，表示层应该是对于数据访问层是“无知”的，这样既减少了层与层间的依赖关系，也能有效避免“循环依赖”的后果。

本系统中，抽象出来的 IDAL 模块，除了解除了向下的依赖之外，对于其上的业务逻辑层，同样仅存在弱依赖关系，如下图所示：

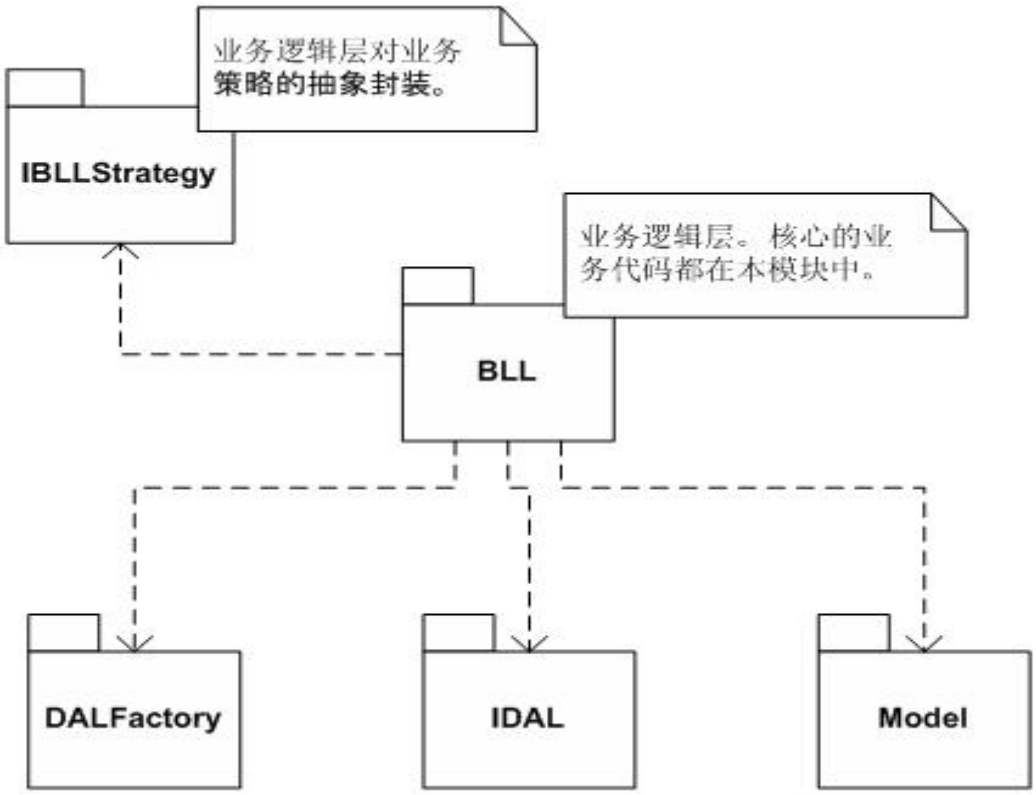


图 3.10 业务逻辑图

上图中 BLL 是业务逻辑层的核心模块，它包含了整个系统的核心业务。在业务逻辑层中，不能直接访问数据库，而必须通过数据访问层。注意图中对数据访问业务的调用，是通过接口模块 IDAL 来完成的。既然与具体的数据访问逻辑无关，则层与层之间的关系就是松散耦合的。如果此时需要修改数据访问层的具体实现，只要不涉及到 IDAL 的接口定义，那么业务逻辑层就不会受到任何影响。毕竟，具体实现的 SQLServerDAL 和 OracalDAL 根本就与业务逻辑层没有半点关系。

3.4 本章小结

“金无足赤，人无完人”，分层式结构也不可避免具有一些缺陷：

1、降低了系统的性能。这是不言而喻的。如果不采用分层式结构，很多业务可以直接造访数据库，以此获取相应的数据，如今却必须通过中间层来完成。但当今由于硬件功能的强大可以完全解决此问题。

2、有时会导致级联的修改。这种修改尤其体现在自上而下的方向。如果在表示层中需要增加一个功能，为保证其设计符合分层式结构，可能需要在相应的业务逻辑层和数据访问层中都增加相应的代码。

当然我们也必须看到它独具的优势所在：

1、开发人员可以只关注整个结构中的其中某一层；

2、可以很容易的用新的实现来替换原有层次的实现；

3、可以降低层与层之间的依赖；

4、有利于标准化；

5、利于各层逻辑的复用。

综所上述，由于杭州电视台独特的业务需求，故上述的系统架构和系统设计模式是为其量身打造，是适合目前电视目前需求的最佳系统架构。

第 4 章 系统模块划分及数据库设计

4.1 系统模块划分

目前此系统最主要的业务是针对电视各频道栏目下的新闻报料归类整合，所以将系统分为频道管理，报料管理，用户管理和系统设置四大模块。如图：

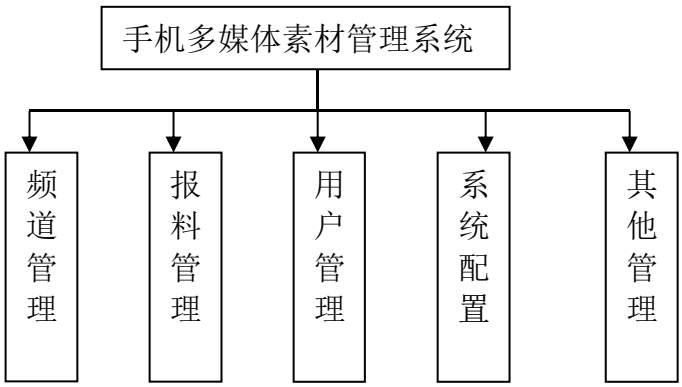


图 4.1 模块划分图

具体介绍如下：

（1）频道管理模块

此模块主要负责各频道的创建，编辑和频道下各栏目的创建，编辑。电视台的新闻都是以频道和栏目为主体进行管理的。

（2）报料管理模块

此模块包含报料管理和评论管理，报料管理负责对手机多媒体素材的种类归类以及各种详细的操作，该系统的所有业务核心在此模块中，包括手前台手机 WAP 网站的支持等。评论管理主要针对手机 WAP 页面新闻的评论管理，由于电视台政治觉悟性的高度， 故此模块需有详细的审核等操作。

（3）用户管理模块

此模块包含两大管理，一是对即将开发的前台 WAP 页面的用户管理和系统本身后台系统的权限管理，前台用户管理这里简单划分为注册和为注册用户，注册即可评论。后台角色管理，普通管理员可以进行频道栏目的新建编辑，新闻的管理审核等操作，对角色管理和系统配置则没有权限，超级管理员即后台的最高权

限，可以查看，操作系统的一切功能。

（4）系统配置模块

此模块主要有系统配置的一些信息和数据库管理，系统配置信息简单的列举了系统文件存放的一些物理路径和一些文件的格式，而数据库管理主要分为备份和还原，因为数据对电视台的重要性，我们必须要做到万无一失，此功能可以每天进行人工备份，以防将来可能出现的数据丢失。

（5）其他信息

这里主要指系统显示当前日期，当前用户，以及当前用户对自己的便捷操作和一些帮助信息。

4.2 数据库设计

数据库系统是信息系统支撑系统的重要组成部分，用面向对象分析与设计方法最终得到的系统对象模型与 OODB 相吻合[19]。数据库设计的要求是建立一个结构分明，逻辑体系严谨的相对独立的数据库体系，在功能、性能上满足手机多媒体素材管理系统处理的要求，同时提供最大限度的对多媒体素材处理决策的支持。

数据库设计的基本原则是在系统总体信息方案的指导下，个人数据库应当为它的各个用户管理目标服务。在设计数据库时，应重点考虑一下几个因素[20]：

(1)数据库必须层次分明，分布合理。

(2)数据库必须高度结构化，保证数据的结构化、规范化和标准化，这是建立数据库和进行信息交换的基础。数据结构的设计应该遵循国家标准和行业标准，尤其要重视编码的应用。

(3)在设计数据库的时候，一方面要尽可能地减少冗余度，减小存储空间的占用，降低数据一致性问题发生的可能性，另一方面，还要考虑适当的冗余，以提高运行速度和降低开发难度。如表 4.1—4.7 表，这里把主要子模块的表列举出来，其他详见数据库。

表 4.1 频道表

字段名称	含义	数据类型	长度
ID	主键 ID	int	整型
SiteID	频道 ID	int	整型
SiteName	频道名称	nvarchar	50

Intro	简介说明	nvarchar	500
ServerIndex	IIS 频道索引	int	
ServerComment	IIS 频道名称	nvarchar	50
ServerDomain	IIS 频道域名	nvarchar	50
CreateUserID	创建者 ID	int	整型
CreateUserName	创建者名	nvarchar	50
LastEditUserID	最后修改者 ID	int	整型
LastEditUserName	最后修改者名	nvarchar	50
CreateTime	创建时间	smalldatetime	
PublishMessage	发布信息说明	nvarchar	200
PublishTime	发布时间	smalldatetime	
PublishStatus	发布状态	tinyint	
Status	系统状态	tinyint	

表 4.2 栏目表

字段名称	含义	数据类型	长度
ID	主键 ID	int	整型
SiteID	频道 ID	int	整型
ParentClassID	父分类 ID	nvarchar	50
Title	栏目名称	nvarchar	50
Intro	简介说明	nvarchar	500
ClassType	分类类型	tinyint	
OrderBy	排序	tinyint	
CreateUserID	创建者 ID	int	整型
CreateUserName	创建者名	nvarchar	50
LastEditUserID	最后修改者 ID	int	整型
LastEditUserName	最后修改者名	nvarchar	50
CreateTime	创建时间	smalldatetime	
LastEditTime	最后编辑时间	smalldatetime	
Levels	发布时间	int	
IsSpePage	是否特殊化页面	tinyint	

Status	系统状态	tinyint	

表 4.3 新闻表

字段名称	含义	数据类型	长度
ID	主键 ID	int	整型
SiteID	频道 ID	int	整型
Title	新闻标题	nvarchar	50
ThumbImageUrl	缩略图路径	nvarchar	500
FileFolderPath	存储文件夹路径	nvarchar	500
Content	新闻内容	ntext	
Author	作者	nvarchar	50
Source	来源	nvarchar	50
ParentClassID	父类 ID	int	整型
Flags	标志(推荐,滚动,热点,头条)0000	nvarchar	50
Tags	新闻标签(以|分割)	nvarchar	100
Views	浏览次数	int	
Rate	评分总分	int	
RateTimes	评分次数	tyint	
CreateUserID	创建者 ID	int	整型
CreateUserName	创建者名	nvarchar	50
LastEditUserID	最后修改者 ID	int	整型
LastEditUserName	最后修改者名	nvarchar	50
CreateTime	创建时间	smalldatetime	
LastEditTime	最后编辑时间	smalldatetime	
IsRef	引用类型，0 没有引用，1 被其他站点或分类引用，2 该项是引用的,3 是该项是复制过来的	tinyint	
RefLink	引用链接	nvarchar	500

Status	系统状态	tinyint	
ShortTitle	短标题	nvarchar	100
GoodsCount	好评数	int	

表 4.4 评论表

字段名称	含义	数据类型	长度
ID	主键 ID	int	
SiteID	频道 ID	int	
ObjectID	新闻 ID	int	
Content	评论内容	nvarchar	500
CreateUserID	创建用户 ID	int	
CreateUserName	创建用户名	nvarchar	50
CreateIP	创建 IP	nvarchar	20
CreateTime	创建时间	smalldatetime	
Status	状态	tinyint	

表 4.5 用户表

字段名称	含义	数据类型	长度
UID	用户 ID	int	
SiteID	频道 ID		
UserName	用户名	nchar	20
NickName	昵称	nchar	20
Password	密码	char	30
Email	邮件地址	char	50
BDay	生日	char	10
ShowEmail	是否显示邮箱	nvarchar	
OnlineState	在线状态, 1 为在线, 0 为不在线	int	
Rights	权限(附加)	nvarchar	100
RightLevel	权限级别(排序)	money	

Status	状态	int	

表 4.6 权限表

字段名称	含义	数据类型	长度
RID	权限 ID	int	
RightTitle	权限描述	nvarchar	100
Rights	权限	nvarchar	100
RightLevel	权限级别(排序)	money	
CreateTime	添加时间	smalldatetime	
Status	系统状态	tinyint	

表 4.7 权限补充表

字段名称	含义	数据类型	长度
RUID	权限用户 ID	int	
UID	用户 ID	int	
RID	权限 ID	int	

此外本项目系统配置均写在 WebConfing 文件里，不采用数据库模式，系统文件存储目录的更改和数据库的还原备份直接对配置文件进行操作。

第 5 章 系统功能模块业务分析

5.1 系统业务概要

本系统从业务层上分为：频道管理，报料系统，用户管理和系统配置四个业务子系统。用户登录本后台系统后可以一目了然上述四个子系统，并根据自己的权限大小以及自己的需求进行相应操作。当然我们也可以在后期可以在此系统上平行增加需要的子系统。

5.1.1 频道管理业务分析

频道管理系统主要管理各频道与各栏目的新建，编辑，删除，修改等操作，栏目不会单独存在，而是依附频道而存在，在树形目录下各频道并列存在，栏目作为频道二级目录而存在。如图 3.1：

图 3.1 频道列表图

当用户登录后的页面会默认为频道列表的第一个频道所对应的对内容，所以其栏目一栏下面也对应的是该频道下的各栏目。如果选择其他频道的话，则可以在频道列表的下拉菜单里找到对应的频道，如果没有则自己可以选择创建频道，如图 3.2：

图 3.2 创建频道图

创建完后就可以在频道列表的下拉菜单里找到所对应的频道，当然如果输入频道信息错误的话则可以点击编辑频道进行修改，同样在栏目列表里也可以根据一个频道多栏目需求进行相应的新建、编辑或者删除等操作如下图：

图 3.3 创建栏目图

我们可以在新建栏目的下面看到栏目子列表的部分显示，即已经创建过了的栏目会根据创建时间倒序排列，方便我们修改，编辑和删除等操作，最后无论是创建好的频道还是创建好的栏目都需要审核，由于电视台政治因素比较强烈的原因，所有的频道或者栏目创建都需要审核通过才可以进行其他操作，否则便一直处于闲置无操作状态。

5.1.2 报料系统业务分析

报料系统可以分为两部分，报料管理和评论管理，而报料管理是本项目的核心，通过选题背景可以知道本项目就是为管理整个报料新闻而服务的下面我们先看下整个新闻报料的界面:如下图 3.4

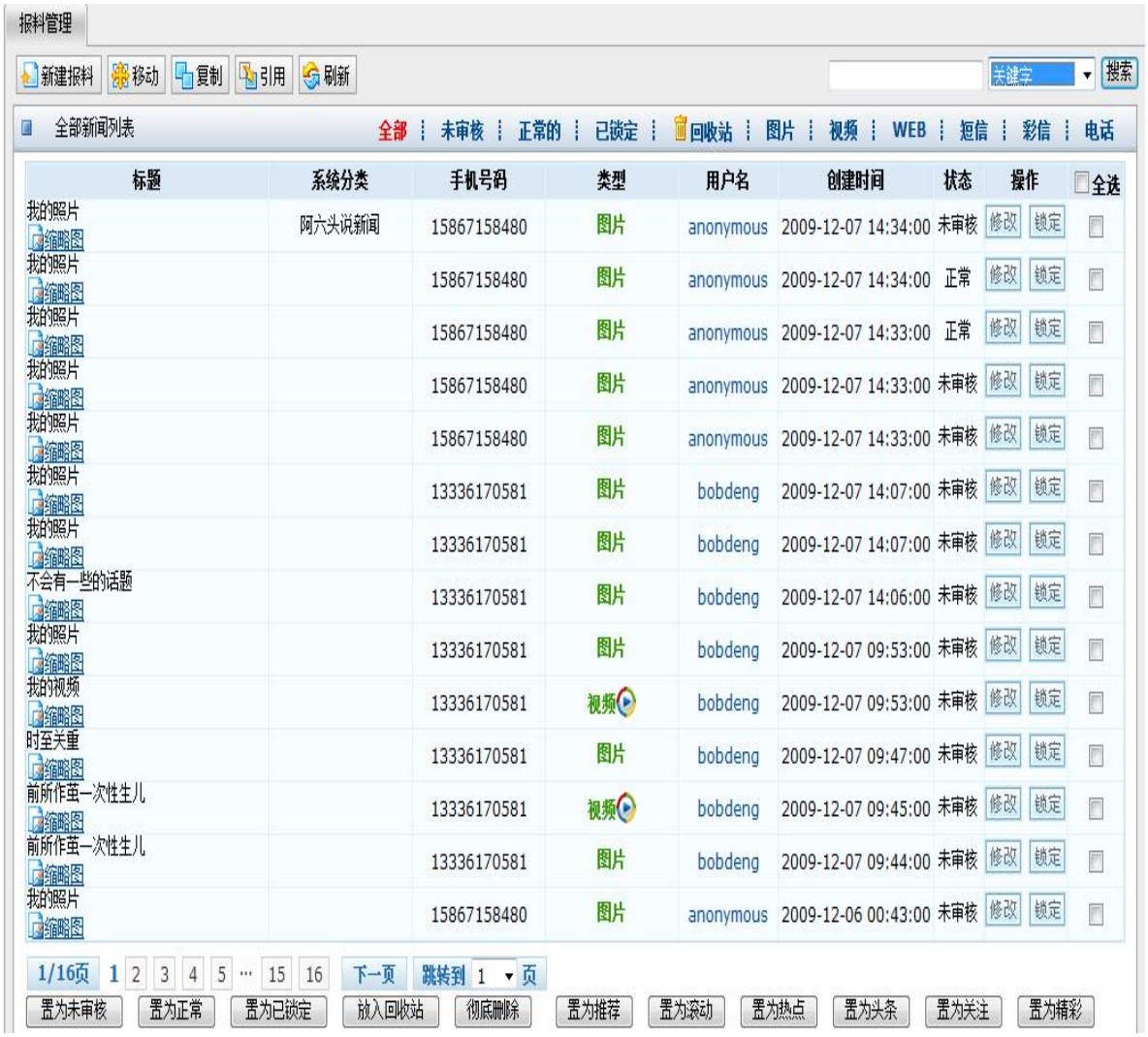


图 3.4 报料界面图

从上述界面可以看到报料得到的多媒体素材可以分为全部、未审核、正常和已锁定四种状态，全部是指全部的多媒体报料素材，点击此按钮下面页面会显示全部的多媒体素材，未审核即不能播出或者推送的多媒体素材，点击此按钮下面页面则会显示未审核的多媒体素材，正常的则是指经过审核可以在频道或者栏目里播出的多媒体素材，点击此按钮则可以获取正常的多媒体素材列表，已锁定主要是指针对某些特定素材一个人可能把握不准确，需要经过上级或者其他人员再次进行审核的，先做好标记，点击此按钮可以获取锁定的多媒体素材列表。回收站这里会存放一些被删除的多媒体素材，在回收站的右端，我们还可以看到图片，视频，WEB，短信，彩信，电话等不同形式的多媒体素材。在页面的下方，我们可

以看到置为未审核，置为正常，置为已锁定，放入回收站，彻底删除，置为推荐，置为滚动，置为热点，置为头条，置为关注等操作，下面对每一操作做详细的介绍：

置为未审核：将选取的多媒体素材改变为未审核的状态。

置为正常：将选取的多媒体素材改变为正常的状态。

置为已锁定：将选取的多媒体素材改变为已锁定的状态。此外我们也可以直接在需要锁定的内容行上点击锁定按钮，这样更加方便和快捷。

放入回收站：将选取的多媒体素材改变为删除的状态，这里的回收站相当于我们 Windowns XP 系统回收站的功能，还具有可以恢复性。

彻底删除：此功能只能在回收站里进行，即将回收站的材料进行彻底删除，不能再将其恢复。

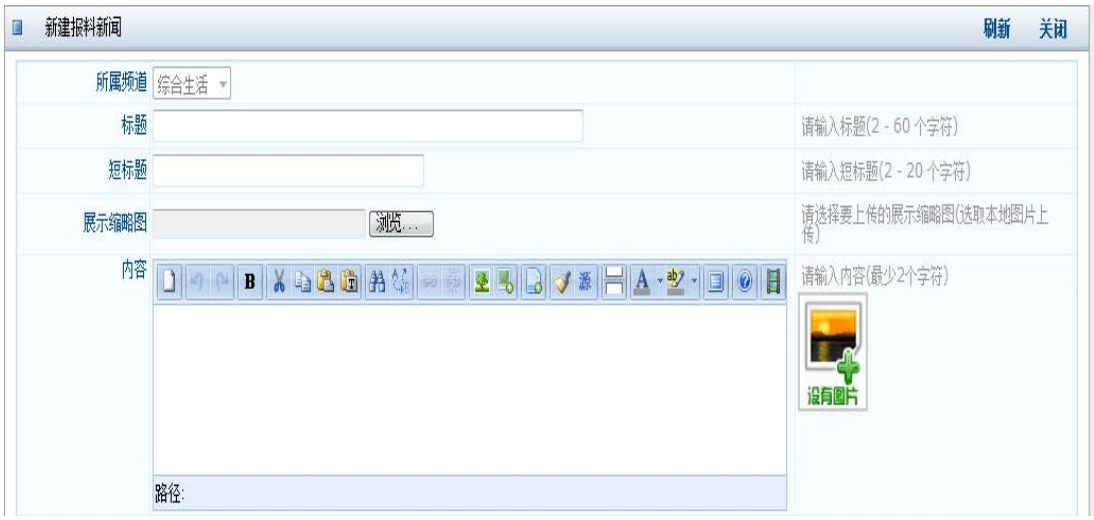
置为推荐：这里支持将来的前台 WAP 页面推荐部门的显示，处于此状态，前台 WAP 页面将会在推荐栏目显示为最前面。

置为滚动：对选取的素材点击此操作，此素材会在前台滚动一栏出现滚动。

置为热点：对选取的素材进行此操作，此素材会在对应的热点栏目前列出现。

置为关注：对选取的素材进行此操作，此素材会在对应的关注栏目出现。、

在此页面的顶端，还有新建报料，移动、复制、引用、刷新和搜索等功能。如下图：



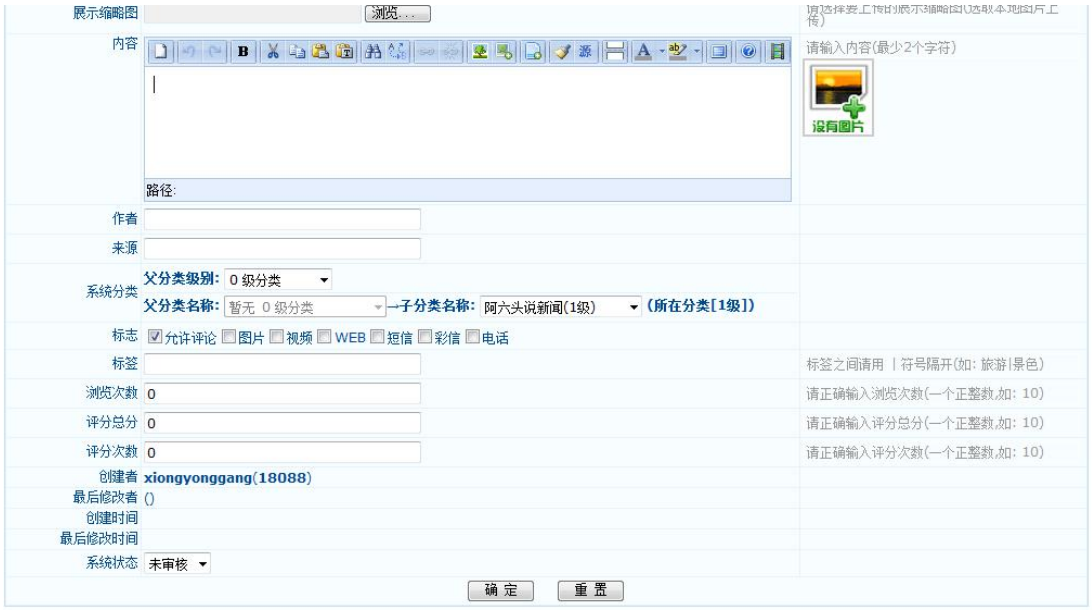


图 3.5 修改报料界面图

新建报料内容分为所属频道，标题，短标题，展示缩略图，内容，作者，来源，系统分类，子分类，标志，标签，浏览次数，评分总分，评分次数，创建者，最后修改者，创建时间，最后修改时间和系统状态。其中内容部分我们引用了一个比较强大的 HTML 内容编辑控件。如果需要对报料内容进行编辑，则点击每个报料内容的修改操作即可，修改获取的页面与新建报料的页面一样。

移动：指将栏目素材移动到别的频道或者栏目下面，原来的频道或者栏目不在有此报料素材。

复制：是指可以将此栏目下的素材复制到一个或者多个别的频道栏目下，其来源属性也显示为别的频道栏目。

引用：指一个或者多个

报料系统里除了报料管理栏目外还有评论管理一栏，这里仅做下简单介绍如下图：

图 3.6 评论页面图

因为将来前台可能要开放用户浏览此节目的留言评论之类的，所以也需要同报料相等的操作，评论有特定的默认排序，所以不需要推荐，精彩，关注等操作。

5.1.3 用户管理业务分析

用户管理主要分为两部分前台 WAP 网站用户的普通管理和后台用户的角色管理。如下图：

图 3.7 用户管理图

由于本项目暂时目标主要是在后台为频道服务,兼顾前台 WAP 页面网站管理，所以对前台 WAP 页面用户权限划分不是很严密，用户注册成为普通用户，便可以留言或者评论，当然后台里面也可以手动添加。此外后台还开通以下一些操作用于管理前台用户，如下图：

图 3.8 用户功能图

角色管理主要在后台分为普通用户和超级管理员，普通用户的权限等级仅对

于频道管理和报料管理两个子系统。而超级管理员的权限等级在所有子系统之上，如下图：

图 3.9 角色管理员图

这里每一名用户还是管理员或者超级管理员，都会显示详细的注册信息，如果需要新建用户业务操作如下图：

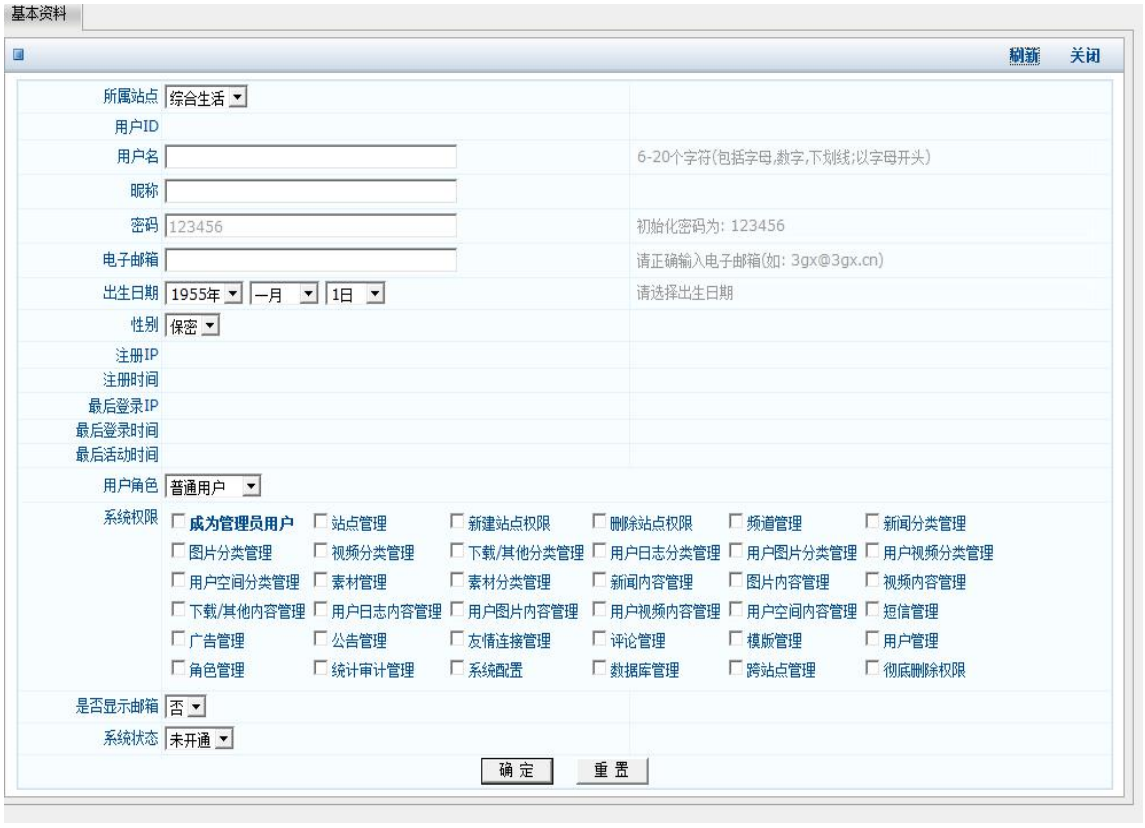


图 3.10 创建角色图

5.1.4 系统配置业务分析

系统配置业务主要有系统配置还有数据库管理，其中系统配置分为目录配置

和其他配置，如下图：

图 3.11 系统配置图

目录设置里主要设置系统多媒体素材存放的一些物理路径，比如：视频和图片这里数据库里存入的只是物理地址，这样当数据量过大时可以减少数据库的压力，我们取相应的多媒体素材只需点击相应的素材进行下载即可。如下图：

类型	路径	说明
存储文件网络	http://sc.hcrt.cn/wapcms/	系统生成和上传文件的网络路径
存储文件根目录	E:\Downloads\wapcms\Upload\	系统生成和上传文件的根目录(硬盘路径)
存储文件根目录的相对路径	/Upload/	系统生成和上传文件的根目录的网络路径
	编辑器配置	
图片管理器根目录中文名	图片目录	图片管理器树形目录的根目录中文名
图片管理器根目录网络路径	http://sc.hcrt.cn/wapcms/Upload	图片管理器上传目录的网络路径
图片管理器根目录路径	E:\Downloads\wapcms\Upload	图片管理器上传目录的根目录(硬盘路径)
文件管理器根目录中文名	文件目录	文件管理器树形目录的根目录中文名
文件管理器根目录网络路径	http://sc.hcrt.cn/wapcms/Upload	文件管理器上传目录的网络路径
文件管理器根目录路径	E:\Downloads\wapcms\Upload	文件管理器上传目录的根目录(硬盘路径)

图 3.12 目录配置图

其他设置里主要显示的本系统的一些补充信息，可以更加了解系统配置的一些信息，如下图：

图 3.13 其他配置图

考虑到硬盘的损坏或者操作系统出现意外而导致数据的丢失，我们采取每一天都采取数据库备份，数据库备份和数据库还原设置，如下图：

图 3.14 数据库操作图

5.1.5 其他业务分析

这里主要指页面上的附加信息，如下图：

图 3.15 其他业务图

后台页面上会显示出用户登录的用户名，以及当前日期，然后会附加出现一些额外的信息和功能，如退出，我的信息，密码，关于，帮助以及 WAPCMS。退出便返回到登录此系统的页面，我的信息页面如下：

图 3.16 我的信息图

关于，帮助以及 WAPCMS 都是些系统的辅助信息，这里不再过多做详细解释。

5.2 本章小结

本章详细介绍了手机多媒体素材管理系统的业务流程和业务范围，用户可以通过登录此后台系统更加详细体验此后台系统，此外，系统有待完善的地方敬请提出，我们会及时作出更改。

参考文献

[1]刘振岩，基于.NET 的 Web 程序设计—ASP.NET 标准教程[M].北京：电子工业出版社，2006.

[2]周峰，Visual C#.NET 2005 基础与实践教程[MI.北京：电子工业出版社，2002.

[3]闰洪亮,ASP．NET 设计教程[M]上海：上海交通大学出版社，2006.

[4]陈轶,Web 开发技术实用教程[M]北京：清华大学出版社，2008

[5]林义证，Visual C#2005 完全开发指南，北京：科学出版社，2008.

[6]Microsoft.NET 框架程序设计 （修订版）/里克特（Richter.J）著；李建忠译—清华大学出版社，2003（微软.NET 程序员系列）

[7]佚名，.NET Framework-Microsoft Visual Studio.NET 简介，互联网文章

[8] Jesse Garrett.AJAX: A New Approach to Web Applications [EB/OL]. http://www.adaptivepath.com/publications/essays/archives/000385.php,2005-02-18.

[9]Bemmel JH, Musen MA. Handbook of Medical Information [M].Netherlands: Bohn Staflen Van Loghum, 1998.

[10] J.W. Cooper。C#设计模式.张志华等译，2003 年 8 月.

[11]GOF 著,设计模式—可复用面向对象软件的基础。/李英军等译.机械工业出版社.

[12].NET 与设计模式/甄镭编著—北京：电子工业出版社，2005.6（.NET 技术大系）.

[13]蔡宏伟，金连甫，陈平.可扩展三层 B/S 体系结构研究和应用[J7]，浙江理工大学学报，2006, 23 (2):178-181.

[14]龚波.软件过程管理[M].中国水利水电出版社，2003.

[15]吴鸿雁，刘思源.基于 OO 技术的关系数据库设计[J].计算机工程，2003, 29

[16][荷兰]ImarSpaanjaars，ASP.NET3.5 入门经典，北京：清华大学出版社，2008.

[17]陶勇，HIBERNATEORM 最佳实践，北京：清大学出版社，2007，9

[18] [美]C,亚历山大等著。建筑模式语言。王听度等译。知识产权出版社，2002 年 2 月

[19〕王静，王先培，马方明，王玉华.面向对象分析方法及在管理系统中的应用[J].武汉大学学 报(王学版)，2003

[20]Guvenir H A,Emeksiz N. An Expert System for the Differential Diagnosis of Erythemato-squamous Diseases [J].Expert System with Applications, 2001:18.

作者简历

教育经历：

2004.9-2008.6 就读于浙江万里学院 计算机科学与技术

2008.9-2010.6 就读于浙江大学软件学院 应用软件开发

工作经历：

2009 年 5 月至 2010 年 2 月在杭州市文化广播电视集团科研所实习

2010 年 3 月至今在杭州市文化广播电视集团影视频道

攻读学位期间发表的论文和完成的工作简历：

致谢

署名

于浙江大学软件学院

当前日期