学生姓名： 韦小龙

用户名： 17weixiaolong

所属教学服务中心： 广州电大教学服务中心

指导教师：

提纲：

本文从基于B/S的企业管理系统设计中的企业办公自动化系统，英文为Office Automatin，简称OA系统的开发进行深入的阐述，此次开发的OA系统是采用主流的B/S结构，开发框架为.NET。本文将会从系统产生的背景，能解决什么问题，系统功能的设计，系统框架的设计，采用的编程语言，开发工具，开发环境，数据库的设计等方面进行一步一步的描述。清楚的记录了一个系统从设计到开发完成的历程，开发过程中会对OA系统中常用功能的实现和核心模块功能进行详细的解决方案分析和技术的实现。

1绪论

1.1研究背景

1.2国内外现状

1.3研究内容

1.4本章小结

2关键技术介绍

2.1B/S架构

2.2数据库技术

2.3系统开发模式

2.4 本章小结

3企业管理系统需求分析

3.1可行性分析

3.1.1投资必要性

3.1.2经济可行性

3.1.3技术可行性

3.2业务流程分析

3.3功能模块分析

3.4本章小结

4企业管理系统设计及功能实现

4.1系统技术架构设计

4.2功能模块设计

4.2.1登录设计

4.2.2用户管理设计

4.2.3角色管理设计

4.3功能模块实现

4.3.1登录模块实现

4.3.2用户管理实现

4.3.3角色管理实现

4.4数据库系统设计

4.4.1数据库表的设计

4.5本章小结

5企业管理系统软件测试

5.1测试环境及数据

5.2功能模块测试及分析

5.3本章小结

6结论和展望

6.1结论

6.2展望

参考文献

**企业管理系统设计**

【摘要】：

  随着社会的不断发展，企业进入了网络时代，信息化需求大大增加，传统的办公模式已经极大地束缚了人的创造力和想象力。领导在日常繁忙的工作中，不能及时、有效的对各项工作按轻重缓急来处理，给许多领导带来了繁重的工作压力。用先进的、现代化的工具代替手工作业，无疑是生产力发展的方向。企业管理系统对传统办公方式的变革，正是适应了人们的普遍需求，也顺应了技术发展的潮流，自然成为业界追求的目标。

本文从企业管理系统的需求、设计、架构等方面，对系统的功能进行业务分析和对系统功能的开发所用到的技术和理论作深一步的解析。本系统的架构设计方式采用的是当前流行的B/S架构设计，C#语言作为后端开发语言，ASP.NET MVC技术作为平台架构，IIS作为Web服务器，数据库采用SQLServer,另外采用Vsual Studio目前最流行的Windows平台应用程序的集成开发环境组成良好的开发和运行的组合，让系统更高效的运行。

【关键词】：管理系统、OA系统、ASP.NET MVC、Vsual Studio、SQLServer

【正文】：

1绪论

1.1研究背景

外部环境发展的需要，当前社会经济的快速发展，使企事业单位都面对国内外同行业的直接竞争与挑战，为更好地应对挑战，急需以信息技术为支撑点，以信息化为突破口，通过建设OA办公自动化等系统推动单位的业务流程再造与组织结构重组以提升竞争力。这种通过信息化求生存谋发展的迫切需要，与每个行业面临的生存与发展挑战是分不开的。

早在进入21世纪之初，国家就提出了“信息化带动工业化”这一战略方针。逐步使信息技术和信息化渗透到各行各业，加速各产业、产品的结构调整，完善企事业单位的管理体制，满足市场和消费者的需求，提高整体竞争实力。

 企事业单位内部发展的需要，如何实现资源的优化组合使“1＋1＞2，这是当前企事业单位面临的主要问题，实现内部资源互补、资源融合、资源协调、资源统一，从而实现战略目标上的实现。要注重的是不同单位文化的融合、品牌的集中，不但要整合行业内资源，还要完成单位内部业务流程重组与再造、建立科学高效的管理方式。

信息化是目前企事业单位快速发展的必由之路，即“在信息的空间重建企事业单位的运作”——将改革、发展规划与信息化建设充分结合起来，利用信息化手段建立改革、管理、运营支撑体系，辅助推进和落实改革的各项任务，使横跨全省、业务遍布全国的单位成为一个完整、流畅的管控一体化的系统。

企业OA系统建议目标，协同办公OA系统应实现单位内部协作、沟通、规范的应用系统，从而提高办公效率；按照“一套系统、一个平台、一个门户、多级应用”的总体建设思路，构建一个基于互联网的、电子一体化的、覆盖整个组织的信息产业的、统一的现代化办公协同OA平台和信息门户，满足企事业单位跨时间、跨区域、跨部门的集团化协同办公要求；根据不同的业务，建立分类的知识库，形成的知识地图，将人员、流程和业务知识紧密集成在一起，促进信息的有效利用和科技创新，协同办公OA系统应提供对各个业务环节进行整合的方案，建设一个可以面向整个业务过程来进行管理的协作平台。

从应用的层面看，协同办公OA平台以部门之间、跨部门以及企事业单位内部与外部的工作流程带动知识信息流、物流、资金流等在内外的无障碍流动，并完成相关数据库的更新。对于系统本身，任何一个模块都可以看作是系统应用层面上的核心，其他所有的模块都围绕本平台工作，共同完成本平台所要求的相关应用。

从管理的层面上看，协同办公OA平台的设计应基于资源协同，实现“以点带面”和“协同运作”，任何一个因素的变化都在系统中的相关点反应出来，并通过平台提供给企事业单位各部门、以及外部资源，从而使业务过程达到高效、协作的目的。用户只需在统一的平台和界面中便可获得所有的信息及完成必要的操作，而不必在不同的平台、数据库和应用系统之间切换。

1.2国内外现状

OA系统是70年代中期的时候，发达国家用来解决企业中产生的大量办公业务量对企业生产效率产生严重影响的问题，应运而发展起来的一种解决企业管理问题的技术，它借助计算机和网络技术，使人们利用它的实现的各项办公业务功能，大大提升了企业和员工对各种业务的处理效率和办公速度，能够很好的节约公司资源，带来更好的效益。随着OA不断发展，　欧美等发达国家的OA办公自动化管理系统己经发展得非常完善，产品成熟，品牌较多，技术先进，且市场占有率很大，行业经验丰富。发达国家己进入第三阶段的发展，这种发展主要是以知识管理为核心，拓展系统内涵和其标准化的建设。

OA系统是在七十年代末八十年代初进入我国，到现在近三十年的发展历史，在我国国民经济水平的快速提升，计算机技术和互联网的快速发展下，我国的OA系统和其它的企业管理系统都已展发展为世界先进水平，特别是以知识管理为核心的第三代OA系统。

1.3研究内容

为了方便管理员工和提交办公效率，因此系统要统一标准进而规范员工的日工作习惯，提高企业员工的执行力，能够让员工主动的参与公司的业务流程管理，规范业务表转化、模块化并加快业务流转的速度，真正做到“流程统一、责权分明、管理高效”，建立统一的信息发布平台，因为使用用户的信息水平知识高低不一，所以系统一定要简明，易用。

首先，用户根据自己的账号登录进到后台管理页面，只能看到自己职责范围内的内容，管理员可以根据不同员工设定不同的角色，而限制其在管理系统中所能操作的权限，做到责权分明。让行政管理更加简单化，像办公用品管理，与我们日常事务性的办公工作相结合的各种辅助办公，实现了这些辅助办公的自动化，管理流程化，只要划定了权限，每个人都可以了解具体日常工作的进展情况。

1.4本章小结

本章主要从研究背景、国内外现状、研究内容来阐述了企业使用OA系统的必要性，和OA系统在国民经济快速发展和互联网技术广泛应用现状下的现状。通过上面所述，我们清析了解到，OA系统可以提升单位的日常管理，规范工作流程，在时间和空间上提高内部的沟通与协作能力，实现内部信息共享，提高办公效率，节省成本、降低各种资源的消耗等，提高企业的管理水平和管理效率，是企业发展过程中必不可以的工具。

2关键技术介绍

2.1B/S架构

企业管理系统采用的网络结构模式是B/S(浏览器(Browser)/服务器(Server))，是随着Internet技术的兴起后的一种网络结构模式，是对于传统的C/S(客户端（Client)/服务器(Server))方式的一种改进。传统的C/S模式需要下载客户端安装，例如腾讯QQ就一种C/S模式，需要先下载QQ的客户端安装然后才能使用。而B/S只需要安装浏览器，在浏览器上输入要访问的网址就有打开对应的应用。而在windows使用几乎主宰了桌面电脑的市场，而浏览器又是桌面电脑的重要软件之一，windows电脑都会自带了课程器。很明显，B/S架构的的客户端具有更好的通用性，和减少系统开发和维护的成本。

B/S采取浏览器请求，服务器响应的工作模式。用户可通过浏览器去访问Internet上由web服务器产生的文本，数据等信息。而每一个web服务器又可以通过各种方式与数据库连接，所访问到数据实际存放在数据库中，具体的流程如下图所示。

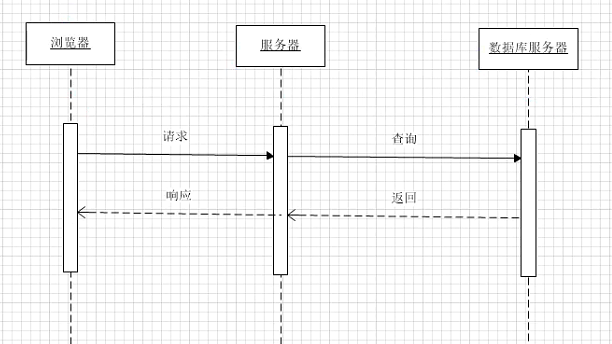


图2-1-1 B/S架构访问流程图

2.2数据库技术

数据库的英文名称叫DB(Database)，就是存储数据的地方，类似于Excel,可把你的数据按录进去然后保存起来，在你想要的时候可以输入特定的条件然后把你想要的数据筛选出来提供使用，数据库在系统中的作用就是把系统的数据都保存在数据库，然后界面上的特定格式的数据就是利用程序从数据库中取出来，经过逻辑处理后显示到系统的界面上供用户查看。

因本系统是OA系统，而OA系统合适采用关系型数据库，本系统采用的是SQLServer数据库技术。SQLServer是[Microsoft](https://baike.baidu.com/item/Microsoft/125917" \t "https://baike.baidu.com/item/Microsoft%20SQL%20Server/_blank) 公司推出的[关系型数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/696511" \t "https://baike.baidu.com/item/Microsoft%20SQL%20Server/_blank)，是目前主流的数据库之一。SQLServer特点有易用性、适合分布式组织的可伸缩性、用于决策支持的数据仓库功能、与许多其他服务器软件紧密关联的集成性、良好的性价比等，同时也为数据管理与分析带来了灵活性，在适合复杂的场景的同时响应速度快，SQLServer是一个具备完Web支持的数据库产品，可以担任这次系统开发的数据库。

2.3系统开发模式

上世纪60年代，由于计算机计算能力和处理的问题复杂度的急速增长，爆发了众所周知的软件危机。为了应对软件危机带来的危害，解决管理大型复杂软件的难题，学术界与工业界共同研究并提出了许多有效的软件开发方法。而其中影响最为深远的，分支最为庞大的方法就是结构化方法。目前，很多软件开发工作组对于结构化方法依旧情有独钟。结构化方法着重强调对现实世界的应用问题进行分解，面向过程设计问题空间的解域，强调自顶向下、逐步求精、模块化设计，这与传统软件开发模式在一定程度上有所契合。结构化分析由结构化分析、结构化设计以及结构化程序设计三部分有机组合而成。

而我们的OA系统因为已经是开发目标清析，各步骤工作内容明确，所以采用结构化方法开发。开发流程为需求分析，概要设计，详细设计，软件开发，软件测试，生产实施。首先清楚了需要实现的功能，然后明确每个阶段的工作内容。注重对开发过程的控制，每个阶段工作完成后，要根据阶段工作目标和要求进行审查，使各阶段工作有条不紊地进行，便于项目管理与控制。开发文档规范化，每个阶段工作完成后，要按照要求完成相应的文档，以保证各个工作阶段的衔接与系统维护工作的便利。在系统分析与设计时，从整体和全局考虑，自顶向下地分解，在系统实现时，根据设计的要求，先编写各个具体的功能模块，然后自底向上逐步实现整个系统。

2.4 本章小结

本章主要介绍了系统开发的关键技术中的B/S架架构原理，数据库技术的简介，和这次系统开发所采用的开发方式和开发流程。

3企业管理系统需求分析

3.1可行性分析

3.1.1投资必要性

对于公司而言，传统的纸质办公给公司带来的阻碍有公司办公耗材浪费，采用无纸化办公方式可以大大节省这方面的开销。传统的通信、文下发、批复、归档，繁多的会议等使办公人员陷入忙乱的工作中，无未能快速提升办公效率和办公质量。还有传统的办公方式很难解决公司多分支机构和下属部门，文件的上传下达、申请、审批都受到了时间的限制，领导异地出差、开会同样会导致公司的一些紧急、重要的文件得不到审批，可能会给公司带来巨大的损失。所以推行OA系统，能给企业的经营者和管理者在行为方式和思维方式上带来革命性进步。随着信息化的加强，传统的决策模式早已不能适应瞬息万变的信息社会，提高中工作效率，加快企业内部信息沟通，领导作为快速准确地决策，这都需要有一个快速有效的办公自动化系统进行一系列有效沟通。推行OA，不仅是管理手段的改善，更重要的是带来管理思想的进步。所以投资建设自己的OA系统是非常有必要性的。

3.1.2经济可行性

本系统开发所用到的开发语言为ASP.NET，数据库为SQLServer，服务器采用Windows系统的IIS，OA系统功能不复杂，开发周期短。再者，OA系统所使用的对象为公司全体员工，使用人员不多，服务器的配置要求不高，所以服务器的支出不会高。然后开发的OA系统功能相对比较简单，不需要太资深的开发人员也是能胜任的，所以开发成本也比较低。相对于我们开发出来的信息系统会对公司带来巨大的帮助，这些成本是微不足道的，所以经济上是可行的。

3.1.3技术可行性

因为公司有自己的技术部门，有自己的开发人员。而且系统的技术选型为.NET技术和SQLServer,开发环境和运行环境都为Windows系统下，很大程度下降低了开发难度。而且公司内部使用的OA系统基本不用考虑并发量和分布式的情况，不需要资深的开发人员，公司是完成有能力独立开发一个自己公司内部的OA系统的。

3.2业务流程分析

这套OA系统业务流程相对比较简单，用户登录后能根据自己所属的部门的岗位只能看到自己职责的内容，做到职责分明。同事能在系统中看到自己的信息，行政人事等能在系统中方便的管理公司事务，尽可能的利用线上解决公司内部的事务。

3.3功能模块分析

OA系统的功能模块应该有：

(1)登录模块。一个系统必不可以的模板，每个用户有自己的用户账号登录进系统用来区分每个人唯一的标识。

(2)系统管理模块。其中应该有菜单管理，用来管理系统中的菜单选项；然后有架构管理，用来管理公司的架构部门和分公司；然后有角色管理，用来标识公司的岗位名称，角色可以指定属于哪一个部门；然后有用户管理，用来管理公司中的员工，用户可以指定属于什么角色的。

(3)权限管理。权限管理是企业信息管理系统的核心模块，因为每个员工的职责不同，所以每个员工登录后只能看到自己职责范围内的功能，不能越权操作。权限管理分为菜单的权限管理，比喻角色为开发工程师的员工登录进去看不到行政管理的“资产管理”页面下面的内容。权限管理还有按钮的权限管理，例如角色为开发工程师和人事都能看到“公司公告”页面，但是开发工程师只有浏览的权限，而人事具有新增，编辑，删除公司的权限，所以角色为人事的用户能看到新增，编辑，删除的按钮，而开发人员看不到。

工作平台。工作平台为大部分员工都需要用到的工能，比喻查看公司公告，知识共享，上班打卡，假期管理等常用的功能。

(4)人事行政管理。因人事和行政有比较多的专职的工作内部，人事行政管理是专门为人事和行政人员设计的功能模块，用于管理公司的事务，功能有资产管理，假期管理，消息通知等。

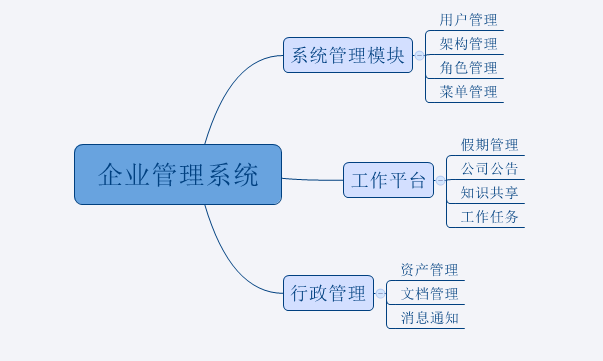


图3-3-1 功能模块设计图

3.4本章小结

本章已从可行性分析说明系统说明了开发本系统的必要性和可行性。也介绍了本系统的业务流程，系统的主要用途。然后也从各个功能模块介绍了系统能干什么事情。通过这章能清楚地了解到这个系统的作用。

4企业管理系统设计及功能实现

4.1系统技术架构设计

软件开发不是千篇一律的流程，我们不能在不了解系统要解决哪个领域的工作的条件下进行软件开发，在开发前，大多数都是需要大量的去做一些业务信息的梳理，然后到软件设计，最后才到开发。而业务信息梳理过程里，我们肯定会形成一个领域知识，根据这个领域知识一步步去驱动软件的设计。本系统就是基于这样的一个思路去做技术设计，这种设计也叫做DDD领域驱动设计。

首先看一下整个项目的架构目录。

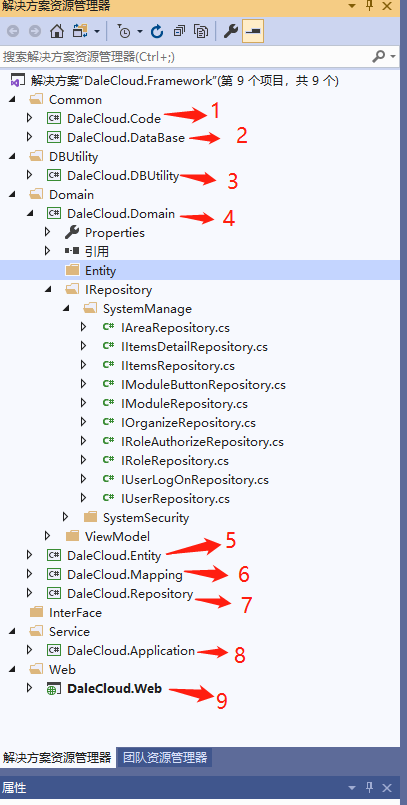


图4-1-1 系统技术架构设计图

Common文件中的类型代表为公共类库，就是项目中各个类型都引用的类库，一些能够整个项目公用的类就会放到这个地方。

编号1的类库的主要为一些扩展类和帮助类，用于辅助项目其实类库实现更简单的一些封装。 编号2的类库为数据库操作的实现，这里用的操作数据库的实现方法为EF(Entity Framework)，一个微软推出的主流的ORM(Object Relational Mapping)框架。简单说一下什么是ORM框架,ORM框架就是框架本身封装了一层访问数据库的方法，能把数据库访问到的数据转为你想要的对象。具体流程如下图

数据库

ADO.NET

ORM

应用程序

图4-1-2 ORM框架访问数据库的流程图

传统的方法是应用程序通过ADO.NET直接访问数据库，现在在应用程序直接通过ORM框架就能访问数据库，而ORM框架在内部其实又会成生ADO.NET的方式自动生成SQL语句来访问数据库，把ORM和ADO.NET的事情都做了。这样使用起来感觉到直接就能取到实体model了，就不需要自己再转换一遍了。

编号3为ADO.NET访问数据库方式，为什么上面已经有了EF的访问数据库方式，下面还要做多一种ADO.NET数据库访问方式呢，因为 EF作为ORM框架更倾向于一些简单的查询，如果是复杂的关联查询的话，ORM框架是根据你写的linq语句来生成sql的，当复杂时生成的语句会不可控，你不清楚你的查询最终生成的sql语句是怎么样的，能不能优化，这个我们都是很难清楚的。所以增加了ADO.NET这种自己手写sql语句的方式查询数据库，这样复杂的查询时如果遇到性能有问题的，我们能把sql在数据库查询界面中把sql语句调优了，再复制到代码里面访问，这样就能做到自主可控。这样简单的数据库操作就能ORM方式访问，复杂的查询就用ADO.NET的方式自己手写语句访问。

编号4为领域层，领域层，作用为把将领域对象从对Repository的依赖中解放出来，避免让领域对象产生直接依赖。里面有接口类，把类的实现定义出来，通过接口直接调用。调用方只需要知道你的接口定义，不需要知道具体的实现。

编号5为存放数据库的实体，里面的每个类对应数据库中的每个表，类里面的字段对应数据库表中的列。

编号6为实体的映射操作，主要作用为把一个对象映射为另一个对象，软件开发中会经常应用到。例如应用程序从数据库中把一个实体查询出来了，但是界面中所使用的字段和这个实体的字段名称不一样，那么就需要把数据库查询出来的实体转换为界面所需要的实体，那么就可以用到这个Mapping类型，把源类转为界面所需要的类。

编号7为仓储层。主要作用为访问数据库， 应用程序需要对数据库操作时，调用这个类库来对数据库进行访问。

编号8为应用层，主要用作一些身份认证之类的属于系统级别的操作。

编辑9为表示层，是程序的主入口，用的是MVC模式。

4.2功能模块设计

4.2.1登录模块设计

用户可以系统登录的界面中输入分配给自己的用户名、密码、登录验证码，点击登录访问系统，系统会先判断用户名、密码、验证码是否为空，然后再判断验证码是否正确，最后判断用户名密码是否正确，如果正确则跳转到系统后台主页，若不正确显示出相应的提示让用户重新登录，算法流程如图4-2-1-1所示。

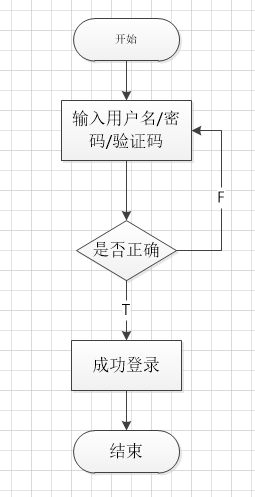


图4-2-1-1 登录流程图

4.2.2用户管理设计

上面说了登录的流程，现在阐述普通员工的账号是哪里来的。就是这个功能模块，用户管理。用户管理的功能就如字面所说的一样，管理用户的账号。主要的功能有新建用户，删除用户，修改用户信息，密码重置，禁用/启用用户，根据输入关键词信息查询符合条件的用户，用户信息用表格展示出来，设置用户有效性和无效。

新增。点击新增按钮弹出编辑窗口，输入用户基本信息，例如账户，密码，姓名，性别，部门，角色等，然后点击确认保存信息，点关闭取消操作。

删除用户。选中用户，点击删除按钮，弹一次再次确认询问，确认删除用户。

修改用户。选中一个用户，点击修改用户，弹出编辑窗，把原用户的信息带进输入框，修改完后，点击确认完成修改。

密码重置。选中一个用户，点击密码重置，弹再次确认框，点确认把用户密码重置为一个固定的密码。

禁用。选中用户，把用户的状态设为禁用。

启用。选中用户，把用户的状态设为启用。

图形关系如图4-2-2-1所示：

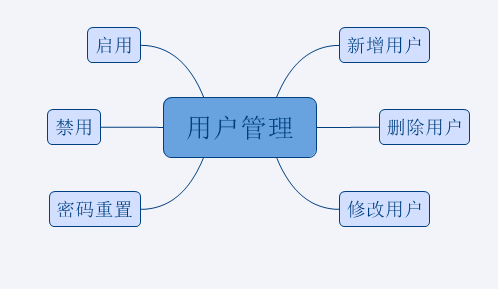


图4-3-1-1 用户管理功能模型

4.2.3角色管理设计

角色管理用于给用户标识一个身份，比喻是管理员，开发人员，人事，行政等。除了给用户职责的标识的同时，也是为了能给角色设置一个权限，用于限制这个角色下的用户所能使用的功能。角色管理模块的功能主要有新增角色，修改角色，删除角色，查询，分配权限，其中权限分配可以分配菜单目录和每个按钮的权限。图形关系如图4-2-3-1所示。

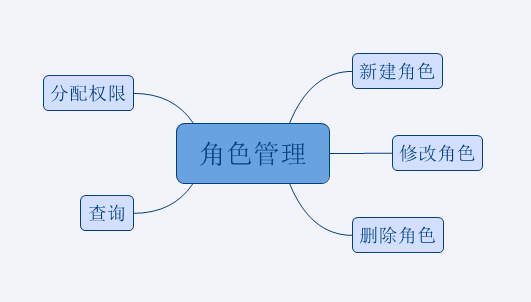


图4-2-3-1 角色管理功能模型

4.3功能模块实现

4.3.1登录模块实现

系统初始化的时候会设定一个超级管理员账号，其它人员的账号可以由这个超级管理员创建或者有权限的用户可以创建新用户。创建的用户信息会存储到用户信息表中。用户打开登录界面的时候，后台会生成一个4位数字和字母组成的验证码，后台预先把显示出来的验证码存到后台的Session中，用户在登录页面输入用户名、密码、验证码后点击提交，前端页面会提前做一些简单的检验，例如用户名，密码，验证码输入框是否为空，如果为空阻止提交操作，及时提醒用户。如果前端通过校验，用户填写的信息会通过表单提交到后台，后台也会先校验表单的合法性，然后对比用户输入的验证码是否和后台Session中保存的验证码一致，若不一致，提示用户验证码不正确。通过了验证码的验证，开始进行用户名和密码的校验，根据用户输入的用户名和密码查询数据库，如果查询不出来，则不存在这个用户，提醒用户用户名或密码错误。其界面设计 如图4-3-1-1所示。



图4-3-1-1 登录界面

关键代码如下：

/// <summary>

/// 生成验证码

/// </summary>

/// <returns></returns>

[HttpGet]

public ActionResult GetAuthCode()

{

return File(new VerifyCode().GetVerifyCode(), @"image/Gif");

}

[HttpPost]

[HandlerAjaxOnly]

public ActionResult CheckLogin(string username, string password, string code)

{

LogEntity logEntity = new LogEntity();

logEntity.F\_ModuleName = "系统登录";

logEntity.F\_Type = DbLogType.Login.ToString();

try

{

if (Session["nfine\_session\_verifycode"].IsEmpty() || Md5.md5(code.ToLower(), 16) != Session["nfine\_session\_verifycode"].ToString())

{

throw new Exception("验证码错误，请重新输入");

}

UserEntity userEntity = new UserApp().CheckLogin(username, password);

if (userEntity != null)

{

OperatorModel operatorModel = new OperatorModel();

operatorModel.UserId = userEntity.F\_Id;

operatorModel.UserCode = userEntity.F\_Account;

operatorModel.UserName = userEntity.F\_RealName;

operatorModel.CompanyId = userEntity.F\_OrganizeId;

operatorModel.DepartmentId = userEntity.F\_DepartmentId;

operatorModel.RoleId = userEntity.F\_RoleId;

operatorModel.LoginIPAddress = Net.Ip;

operatorModel.LoginIPAddressName = Net.GetLocation(operatorModel.LoginIPAddress);

operatorModel.LoginTime = DateTime.Now;

operatorModel.DdUserId = userEntity.F\_DingTalkUserId;

operatorModel.WxOpenId = userEntity.F\_WxOpenId;

operatorModel.LoginToken = DESEncrypt.Encrypt(Guid.NewGuid().ToString());

if (userEntity.F\_Account == "admin")

{

operatorModel.IsSystem = true;

}

else

{

operatorModel.IsSystem = false;

}

operatorModel.IsAdmin = userEntity.F\_IsAdmin.Value;

operatorModel.IsBoss = userEntity.F\_IsBoss.Value;

operatorModel.IsLeaderInDepts = userEntity.F\_IsLeaderInDepts.Value;

operatorModel.IsSenior = userEntity.F\_IsSenior.Value;

OperatorProvider.Provider.AddCurrent(operatorModel);

logEntity.F\_Account = userEntity.F\_Account;

logEntity.F\_NickName = userEntity.F\_RealName;

logEntity.F\_Result = true;

logEntity.F\_Description = "登录成功";

new LogApp().WriteDbLog(logEntity);

}

return Content(new AjaxResult { state = ResultType.success.ToString(), message = "登录成功。" }.ToJson());

}

catch (Exception ex)

{

logEntity.F\_Account = username;

logEntity.F\_NickName = username;

logEntity.F\_Result = false;

logEntity.F\_Description = "登录失败，" + ex.Message;

new LogApp().WriteDbLog(logEntity);

return Content(new AjaxResult { state = ResultType.error.ToString(), message = ex.Message }.ToJson());

}

}

4.3.2用户管理实现

新增。点新增弹出窗口后用户填了输入框后，点提交。前端把用户输入的信息通过Form表单提交到后台，后台接收到数据，对数据校验后保存到数据库，返回操作成功给前端界面。

修改。对用户信息进行修改后，通过表单提交到后台，后台根据当前的用户id更新修改的信息保存到数据库。

删除。提交用户的id,对用户进行数据库删除。

密码重置。提交用户的id，操作数据库更新用户的密码为一个固定的密码。

启用/禁用。提交用户的id,操作数据库更新用户状态为启用/禁用状态。

其界面设计如下图所示。

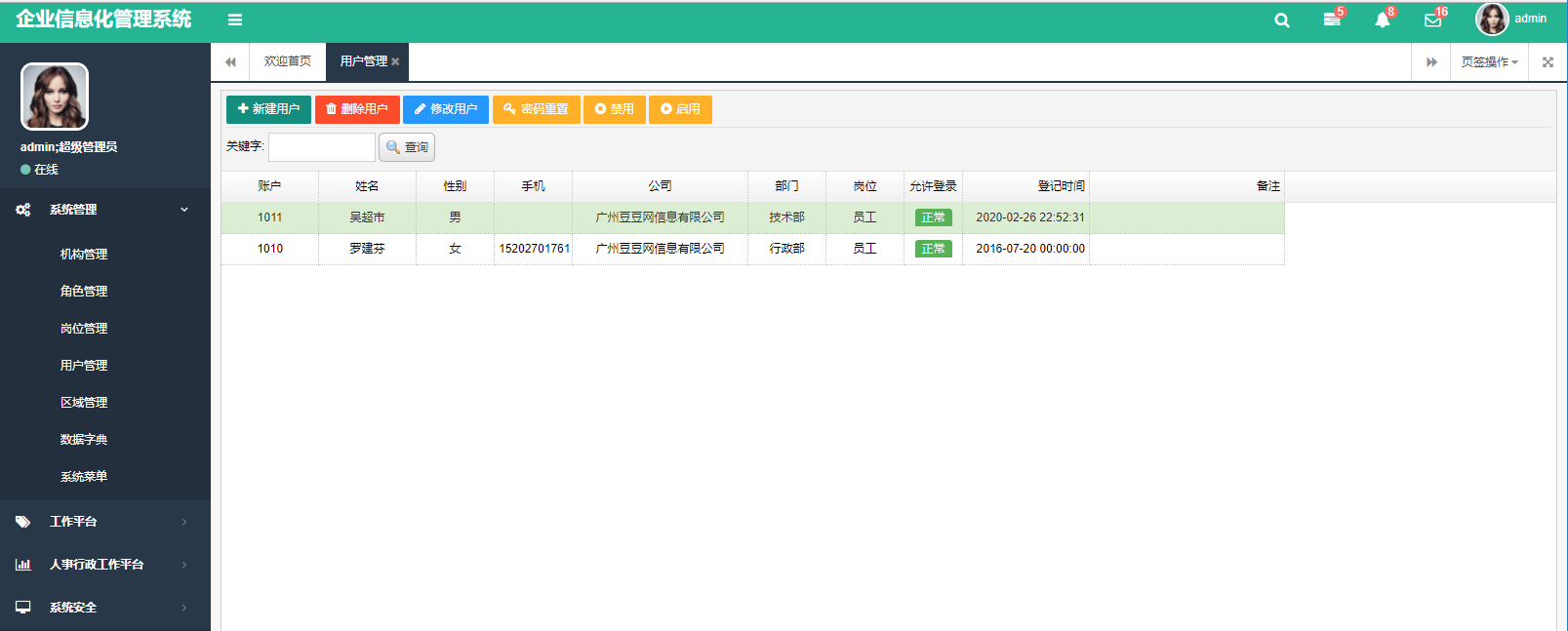


图4-3-2-1 用户管理界面



图4-3-2-2 用户管理新增界面

主要代码如下：

public void SubmitForm(UserEntity userEntity, UserLogOnEntity userLogOnEntity, string keyValue)

{

if (!string.IsNullOrEmpty(keyValue))

{

userEntity.Modify(keyValue);

}

else

{

userEntity.Create();

}

service.SubmitForm(userEntity, userLogOnEntity, keyValue);

}

[HttpPost]

[HandlerAuthorize]

[HandlerAjaxOnly]

[ValidateAntiForgeryToken]

public ActionResult DeleteForm(string keyValue)

{

userApp.DeleteForm(keyValue);

return Success("删除成功。");

}

[HttpPost]

[HandlerAjaxOnly]

[HandlerAuthorize]

[ValidateAntiForgeryToken]

public ActionResult SubmitRevisePassword(string userPassword, string keyValue)

{

userLogOnApp.RevisePassword(userPassword, keyValue);

return Success("重置密码成功。");

}

[HttpPost]

[HandlerAjaxOnly]

[HandlerAuthorize]

[ValidateAntiForgeryToken]

public ActionResult DisabledAccount(string keyValue)

{

UserEntity userEntity = new UserEntity();

userEntity.F\_Id = keyValue;

userEntity.F\_EnabledMark = false;

userApp.UpdateForm(userEntity);

return Success("账户禁用成功。");

}

[HttpPost]

[HandlerAjaxOnly]

[HandlerAuthorize]

[ValidateAntiForgeryToken]

public ActionResult EnabledAccount(string keyValue)

{

UserEntity userEntity = new UserEntity();

userEntity.F\_Id = keyValue;

userEntity.F\_EnabledMark = true;

userApp.UpdateForm(userEntity);

return Success("账户启用成功。");

}

4.3.3角色管理实现

新增。弹出新增窗口，Form表单形式把表单内容提交到后台，保存到数据库。

修改。选中一条角色，弹出修改窗，修改完把内容提交到后台，更新数据库。

删除。选中一个角色，点删除，把角色id提交到后台，对数据库进行删除。

权限分配。修改角色时，弹出一个菜单和按钮的页面，对角色选中相应的权限，把权限信息提交到后台，保存到角色权限表。

界面如下图所示：

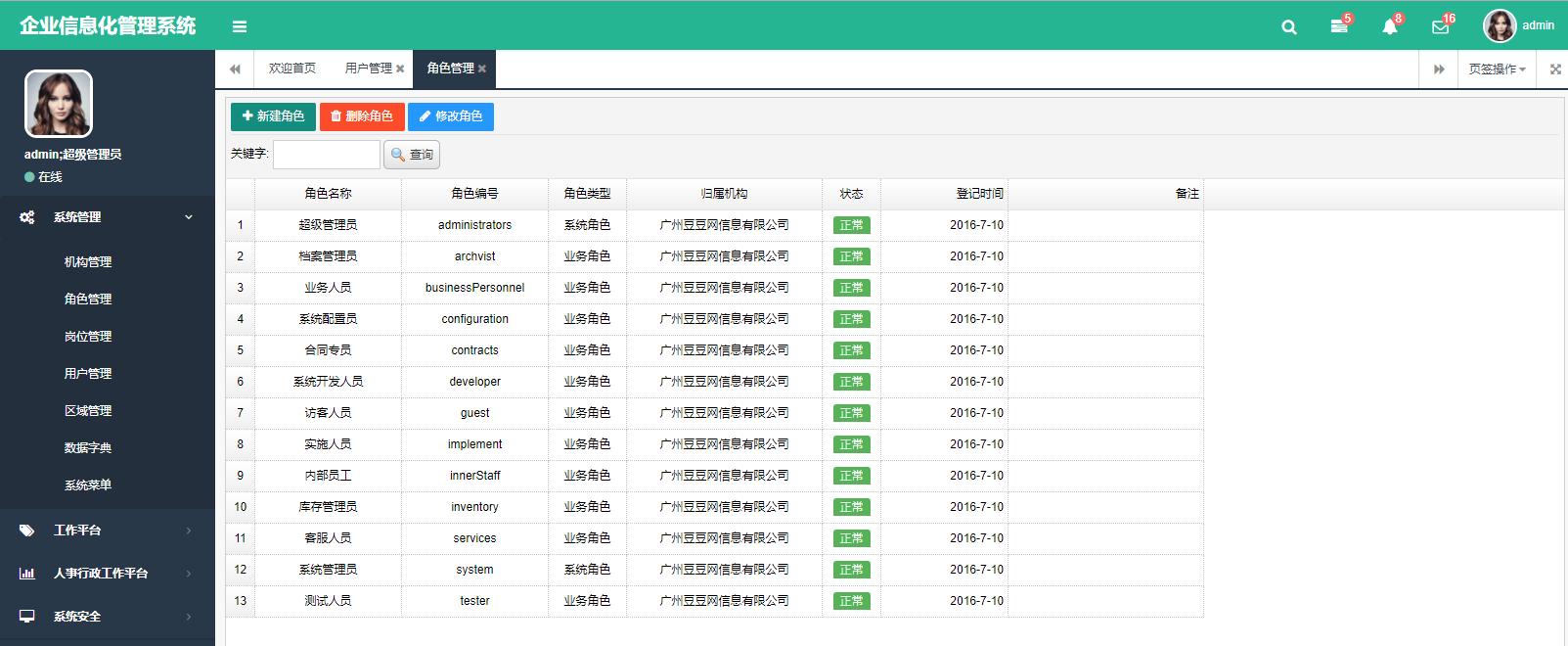


图4-3-3-1 用色管理界面

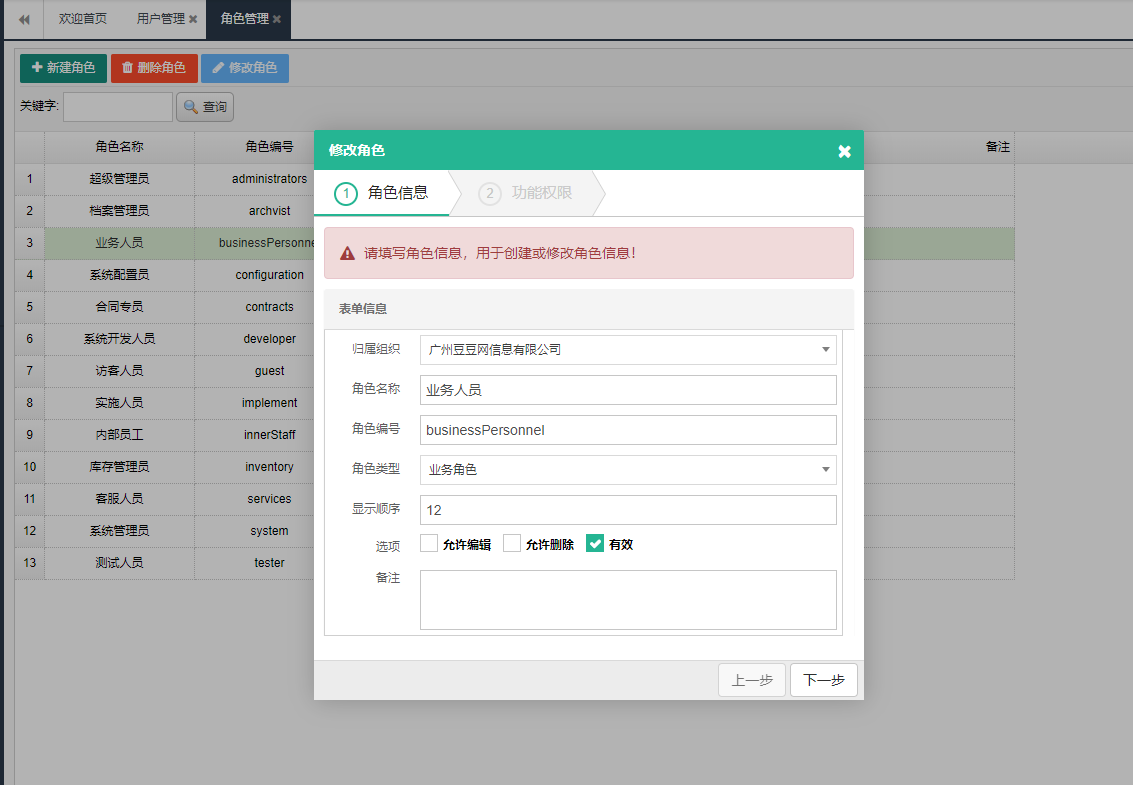


图4-3-3-2 角色修改界面

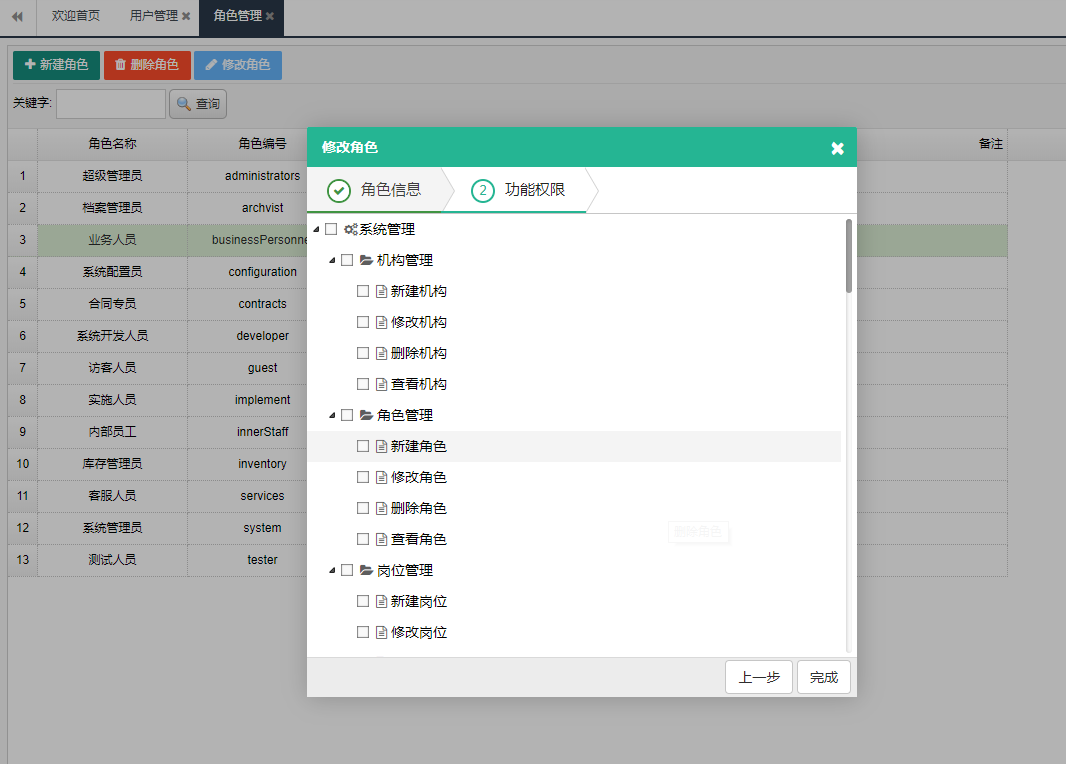


图4-3-3-2 角色权限分配界面

主要代码如下：

public void SubmitForm(RoleEntity roleEntity, string[] permissionIds, string keyValue)

{

if (!string.IsNullOrEmpty(keyValue))

{

roleEntity.F\_Id = keyValue;

}

else

{

roleEntity.F\_Id = Utils.GuId();

}

var moduledata = moduleApp.GetList();

var buttondata = moduleButtonApp.GetList();

List<RoleAuthorizeEntity> roleAuthorizeEntitys = new List<RoleAuthorizeEntity>();

foreach (var itemId in permissionIds)

{

RoleAuthorizeEntity roleAuthorizeEntity = new RoleAuthorizeEntity();

roleAuthorizeEntity.F\_Id = Utils.GuId();

roleAuthorizeEntity.F\_ObjectType = 1;

roleAuthorizeEntity.F\_ObjectId = roleEntity.F\_Id;

roleAuthorizeEntity.F\_ItemId = itemId;

if (moduledata.Find(t => t.F\_Id == itemId) != null)

{

roleAuthorizeEntity.F\_ItemType = 1;

}

if (buttondata.Find(t => t.F\_Id == itemId) != null)

{

roleAuthorizeEntity.F\_ItemType = 2;

}

roleAuthorizeEntitys.Add(roleAuthorizeEntity);

}

service.SubmitForm(roleEntity, roleAuthorizeEntitys, keyValue);

}

public ActionResult GetPermissionTree(string roleId)

{

var moduledata = moduleApp.GetList();

var buttondata = moduleButtonApp.GetList();

var authorizedata = new List<RoleAuthorizeEntity>();

if (!string.IsNullOrEmpty(roleId))

{

authorizedata = roleAuthorizeApp.GetList(roleId);

}

var treeList = new List<TreeViewModel>();

foreach (ModuleEntity item in moduledata)

{

TreeViewModel tree = new TreeViewModel();

bool hasChildren = moduledata.Count(t => t.F\_ParentId == item.F\_Id) == 0 ? false : true;

tree.id = item.F\_Id;

tree.text = item.F\_FullName;

tree.value = item.F\_EnCode;

tree.parentId = item.F\_ParentId;

tree.isexpand = true;

tree.complete = true;

tree.showcheck = true;

tree.checkstate = authorizedata.Count(t => t.F\_ItemId == item.F\_Id);

tree.hasChildren = true;

tree.img = item.F\_Icon == "" ? "" : item.F\_Icon;

treeList.Add(tree);

}

foreach (ModuleButtonEntity item in buttondata)

{

TreeViewModel tree = new TreeViewModel();

bool hasChildren = buttondata.Count(t => t.F\_ParentId == item.F\_Id) == 0 ? false : true;

tree.id = item.F\_Id;

tree.text = item.F\_FullName;

tree.value = item.F\_EnCode;

tree.parentId = item.F\_ParentId == "0" ? item.F\_ModuleId : item.F\_ParentId;

tree.isexpand = true;

tree.complete = true;

tree.showcheck = true;

tree.checkstate = authorizedata.Count(t => t.F\_ItemId == item.F\_Id);

tree.hasChildren = hasChildren;

tree.img = item.F\_Icon == "" ? "" : item.F\_Icon;

treeList.Add(tree);

}

return Content(treeList.TreeViewJson());

}

4.4数据库系统设计

4.4.1数据库表的设计

用户信息表如表4-4-1-1所示：

4-4-1-1 用户信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 长度 | 主键 | 允许空 |
| 1 | F\_Id | Int | 4 | 是 | 否 |
| 2 | F\_Account | varchar | 32 |  | 否 |
| 3 | F\_Password | varchar | 64 |  | 否 |
| 4 | F\_RealName | varchar | 16 |  | 否 |
| 5 | F\_NickName | varchar | 16 |  | 否 |
| 6 | F\_Create\_time | datetime |  |  | 否 |
| 7 | F\_Create\_user | varchar | 16 |  | 否 |

角色信息表如表4-4-1-2所示：

4-4-1-2角色信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 长度 | 主键 | 允许空 |
| 1 | F\_Id | Int | 4 | 是 | 否 |
| 2 | F\_FullName | varchar | 32 |  | 否 |
| 3 | F\_EnCode | int | 4 |  | 否 |
| 5 | F\_Create\_time | datetime |  |  | 否 |
| 6 | F\_Create\_user | varchar | 16 |  | 否 |

菜单信息表如表4-4-1-3所示：

4-4-1-3菜单信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 长度 | 主键 | 允许空 |
| 1 | F\_Id | Int | 4 | 是 | 否 |
| 2 | F\_ParentId | varchar | 32 |  | 否 |
| 3 | F\_EnCode | varchar | 16 |  | 否 |
| 4 | F\_FullName | varchar | 16 |  | 否 |
| 5 | F\_Icon | varchar | 24 |  | 否 |
| 6 | F\_UrlAddress | Varchar | 500 |  | 否 |
| 5 | F\_Create\_time | datetime |  |  | 否 |
| 6 | F\_Create\_user | varchar | 16 |  | 否 |

按钮信息表如表4-4-1-4所 ：

4-4-1-4按钮信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 长度 | 主键 | 允许空 |
| 1 | F\_Id | Int | 4 | 是 | 否 |
| 2 | F\_ModuleId | Int | 4 |  | 否 |
| 3 | F\_FullName | varchar | 16 |  | 否 |
| 4 | F\_EnCode | varchar | 24 |  | 否 |
| 5 | F\_Create\_time | datetime |  |  | 否 |
| 6 | F\_Create\_user | varchar | 16 |  | 否 |