ASP.NET 开发数据库三层架构系统初探

刘 林

摘 要：通过一个新闻发布系统的设计与实现并给出部分代码实例，对基于数据库的三层架构设计进行了初探，并针对设计过程中出现的若干问题进行了分析与解决。

关键词：ASP.NET 技术；C# 语言；数据库；三层架构；VS2008 工具

1 引言

经过国家中职师资培训项目中的网站建设与管理专业的学习，要求最后设计与开发一个基于数据库的新闻发布系统。在开发的过程中，了解三层架构与二层架构的区别，特别是三层架构在后期维护上的的巨大优势。现就 VS2008 下 ASP.NET+C# 环境中三层架构的设计，以及在数据库设计和代码编写等方面得到的经验与教训进行初步探讨。

2 二层与三层

要使用三层模式进行开发，首先要明白其与二层的区别：

（1）二层架构是传统的用户界面（UI）直接访问数据库服务器的模式，以本项目开发环境为例，虽然 ASP.NET 本身已经进行了设计界面（.aspx 文件）与代码页（.cs 文件）的分离，但涉及到数据库，对数据库的连接、操作的定义和实现等，都在同一个.cs 代码页中，增加了程序的复杂性，降低了可维护性。

（2）所谓“三层架构”，即在原来二层架构的用户界面（UI）和数据库服务器之间，人为地加入中间层（即组件层），而中间层又可以划分为业务逻辑层（BLL）、数据访问层（DAL）和数据对象模型层（Model），其中的数据对象模型层（Model）可以把表当做一个对象来处理，充分体现了面向对象的思想。这样，就能使代码部分有效地分层，每层各司其职，降低了程序的复杂性，减少了系统维护的开销和难度。如图 1 所示。

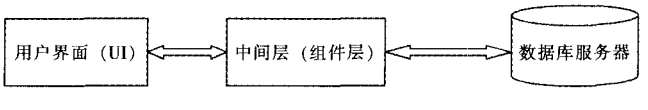

图 1 三层架构模型

（3）三层架构模式因为“高内聚、低耦合”的特性，使其在代码的复用性、灵活性和可维护性上明显优于二层模式，而且由于分层合理，便于多个开发人员之间的协作，每个人只需关注自己负责的那层就行。例如：网站前台设计人员只需遵循事先制定的接口标准，无需了解关于数据库访问的具体技术细节，就可以同数据库等设计人员实现并行开发。

3 系统功能与结构

根据需求，新闻发布系统由前台访问者和后台管理员两大部分组成，前台访问者具有的基本功能：浏览新闻、按照标题或新闻内容在本站内搜索新闻、按照已建好类别进行新闻的分类显示、对某条新闻进行评论；后台管理者具有的基本功能：注册与登录后台管理系统、对新闻的管理（添加、更新、删除）、评论的管理、新闻类别的管理和用户的管理。系统功能结构如图 2 所示。

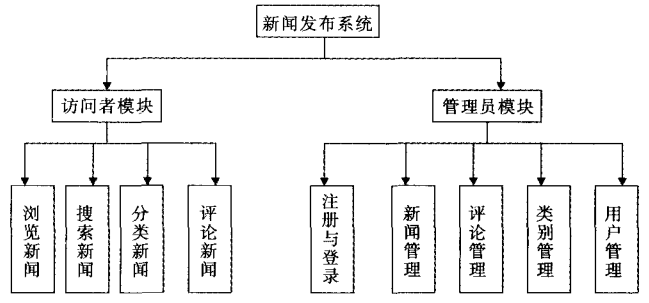

图 2 系统功能结构图

4 数据库

本系统采用的数据库系统为 SQL Server 2005，根据功能分析，建立数据库“newsPub”，由新闻表（news）、新闻分类表（category）、评论表（comment）以及用户信息表（users）构成。如表 1~表 4 所示。

表 1 新闻表（news）结构

列名	数据类型	备注	允许为 null
id	int（自动编号）	新闻编号、主键	No
title	varchar（100）	新闻标题	No
content	image	新闻内容，可存非文本	Yes
createTime	datetime	上传时间	No
caId	int	新闻分类	No
uploader	varchar（50）	上传者	No

表 2 分类表（category）结构

列名	数据类型	备注	允许为 null
id	int（自动编号）	类别编号、主键	No
name	varchar (50)	新闻类别名	No

表 3 评论表（comment）结构

列名	数据类型	备注	允许为 null
id	int（自动编号）	评论编号、主键	No
content	image	评论内容	No
createTime	datetime	评论时间	No
ip	varchar (15)	评论者 ip	No
newsId	int	被评论新闻编号	No

表 4 用户表（users）结构

列名	数据类型	备注	允许为 null
id	int（自动编号）	用户编号、主键	No
name	varchar (20)	注册用户名	No
pwd	varchar (40)	密码	No
email	varchar (50)	用户邮箱	No
uAuthority	int	用户权限	No

5 中间层

5.1 新建空白解决方案

打开 Visual Studio 2008，选择菜单“文件-->新建-->项目-->其他项目类型-->Visual Studio 解决方案-->空白解决方案”，命名为“三层架构新闻发布系统”。

5.2 创建 Model 层（实体层）

（1）添加类库。

在“三层架构新闻发布系统”解决方案上执行“右击-->添加-->新建项目-->Visual C#-->类库”，命名为“Model”。

（2）关于类库的说明。

在 Model 层的设计中，每一个类对应数据库中的一张表，而类中的每个属性对应表中的一个字段（名字不必相同，但要有对应关系）。在“newsPub”数据库中共设计了 4 个表，相应地定义了 4 个实体类 news.cs、category.cs、comment.cs 和 users.cs。

（3）定义实体类（以“News”类为例，其他 3 个实体类的定义类似）。

将生成的“class1.cs”文件重命名为“News.cs”，打开“News.cs”，代码如下：

```
namespace Model
{
public class News
{
  //新闻的 id 字段
 private string id;
 //设置新闻的 id 属性
  public string Id
  {
     get { return id; }
     set { id = value; }
  }
      //新闻的标题字段
  private string title;
 //设置新闻的标题属性
  public string Title
  {
     get { return title; }
     set { title = value; }
  }
 //其他字段与属性定义方法相同(略)
 //空的构造函数
  public News() { }
      //带参数的构造函数 1
   public News (string title, string content, string caid, string uploader)
  {
     //标题等字段的初始化
 this.title = title;
     this.content = content;
     this.caId = caid;
     this.uploader = uploader;
  }
 //带参数的构造函数 2
   public News(string id, string title, string content, string caid, string uploader)
  {
     this.id = id;
     this.title = title;
     this.content = content;
     this.caId = caid;
     this.uploader = uploader;
  }
}
}
```

5.3 创建 DAL 层（数据访问层）

5.3.1 本层作用

该层能够直接操作数据库，包括查找数据、增加记录、更新记录、删除记录等，对应 Model 类库中的每个表类，都有一个相应的 DAL 层访问类。DAL 层能够对它的上一层 BLL（业务逻辑层）提供访问数据库的接口，而又对 BLL 层屏蔽了数据库操作的具体实现。

5.3.2 具体实现

（1）添加“DAL”类库，方法与“Model”类库添加方法相同。

（2）定义“DAL”类库中的类，本系统包括以下5个类：

1）SQLHelper.cs (SQL数据库助手类)。

这个类实现了数据库连接的建立、打开连接、执行参数各异的SQL语句，同时为本层中的其他类提供了参数传递接口，其他类只需提供各种查询所需的参数，就可完成查询，从而进一步提高了公共代码的复用程度。代码如下：

```
//命名空间的引用
using System.Data;
using System.Data.SqlClient;
namespace DAL
{
   public class SQLHelper
   {
      private SqlConnection conn=null;
      private SqlCommand cmd=null;
      private SqlDataReader sdr=null;
      //通过构造函数建立连接
      public SQLHelper()
      {
        //连接字符串根据机器具体情况进行配置
     string strConn = @"server = .server;database =newsPub;
uid=sa;pwd=123456";
         conn = new SqlConnection(strConn);
      }
      //打开连接
      private SqlConnection OpenConn()
      {
        //如果连接关闭,则打开连接
     //ConnectionState 包含在 System.Data 中
      if (conn.State == ConnectionState.Closed)
         {
            conn.Open();
         }
         return conn;
      }
      //执行不带参数的 SQL 语句或存储过程并关闭连接
public int ExecuteNonQuery(string cmdText, CommandType ct)
      {
         int result;
         try
         {
            cmd = new SqlCommand(cmdText, OpenConn());
            cmd.CommandType = ct;
            res = cmd.ExecuteNonQuery();
         }
         catch (Exception ex)
         {
            throw ex;
         }
         finally
         {
            if (conn.State == ConnectionState.Open)
            {
               conn.Close();
            }
         }
         return result;
      }
 //其他数据库操作方法略
   }
}
```

2）NewsDAO.cs (新闻表的数据访问对象类)，代码如下：

```
//命名空间的引用
using System.Data;
using System.Data.SqlClient;
//引用 Model 之前,先要在项目中添加对 Model 的引用
using Model;
namespace DAL
{
   //定义新闻数据访问对象类
     public class NewsDAO
   {
       //通过引用 Model 类库,能够定义 SQLHelper 类型的对
//象 sqlhelper
     private SQLHelper sqlhelper;
      //在构造函数中创建数据库助手类对象,建立连接
      public NewsDAO()
      {
     sqlhelper = new SQLHelper();
      }
      //选择全部新闻方法的定义
      public DataTable SelectAll()
      {
         DataTable dt = new DataTable();
     //查询 news 表中的所有新闻,并按上传时间降序排列
string sql = "select * from news order by createTime desc";
                //调用 SQLHelper 类中的执行查询方法,并传
//递给具体的参数
dt = new SQLHelper().ExecuteQuery(sql, CommandType.Text);
                //返回查询结果,类型是 DataTable 类表对象
         return dt;
      }
//根据新闻 ID 取出新闻内容的方法,调用 SQL 存储过程查询
      public News SelectById(string id)
      {
          //构造 News 类的对象实例 n,作为返回值类型,对应
//数据表中的一条记录
     News n = new News();
     //构造 DataTable 类的对象实例 dt,存储数据库的查询结果
```

```
        DataTable dt = new DataTable();
//预先在 SQL Server 2005 中定义了存储过程"news_selectBy
//Id",并把名字赋给//查询参数 cmdText
        string cmdText = "news_selectById";
      //将参数中的字符串类型的 id 转换为数据库中定义的 id
//类型:int
        int idsql = Convert.ToInt32(id);
        SqlParameter[] paras = new SqlParameter[]{
          new SqlParameter("@id",idsql)
        };
        //传递参数,调用 sqlhelpper 对象执行查询
      dt = sqlhelper.ExecuteQuery (cmdText, paras,
CommandType.StoredProcedure);
    //判断记录中的 content 字段值是否为空
        if (dt.Rows[0]["content"] == DBNull.Value)
        {
          n.Content = "内容暂为空!";
        }
        else
        {
            n.Content = System.Text.Encoding.Default.
GetString(dt.Rows[0]["content"] as byte[]);
        }
        n.Id = id;
        n.Title = dt.Rows[0]["title"].ToString();
        n.CreateTime = dt.Rows[0]["createTime"].ToString();
        n.CaId = dt.Rows[0]["caId"].ToString();
        n.Uploader = dt.Rows[0]["uploader"].ToString();
        return n;
      }
//其他 news 表的操作略
    }
}
```

3) CategoryDAO.cs (类别表的数据访问对象类)。

4) CommentDAO.cs (评论表的数据访问对象类)。

5) UsersDAO.cs (用户表的数据访问对象类)。

5.4 创建 BLL 层 (业务逻辑层)

5.4.1 本层作用

在 DAL 层的支持下，为系统的每个功能模块设计一个类，以实现此模块的业务逻辑，BLL 层是 UI 和 DAL 层之间的桥梁，负责数据参数的传递和处理工作。因为有了底层 DAL 的支持，BLL 层并不涉及具体的数据库访问操作，使逻辑功能更清晰，利于逻辑功能的增删改等变化。

5.4.2 具体实现

(1) 添加 "BLL" 类库，方法与 "Model" 类库添加方法相同。

(2) 定义 "BLL" 类库中的类，本系统包括以下 4 个类：

1) NewsManage.cs (新闻管理业务逻辑类)，代码如下：

```
//命名空间的引用
using System.Data;
//DAL 的引用在整个 BLL 类库中都是必须的,BLL 层通过 DAL
//层中的类的对象实例实现了对 DAL 中方法的调用,同样引用
//前先要在项目中添加对 DAL 的引用。
using DAL;
//Model 的引用根据需要,并不是必须的
using Model;
namespace BLL
{
  public class NewsManage
  {
    private NewsDAO ndao = null;
    //建立与 DAL 层的数据访问接口
    public NewsManage()
    {
      ndao = new NewsDAO();
    }
    //选择全部新闻逻辑方法定义
    public DataTable SelectAll()
    {
          //调用 NewsDAO 类的 SelectAll 方法返回查询
//DataTable 类型的结果
    return ndao.SelectAll();
    }
    //根据新闻 ID 取出该条新闻的具体内容的逻辑方法定义
     public News SelectById(string id)
     {
       return ndao.SelectById(id);
     }
       //其他逻辑方法略
  }
}
```

2) CategoryManage.cs (新闻类别管理业务逻辑类)。

3) CommentManage.cs (评论管理业务逻辑类)。

4) UsersManage.cs (用户管理业务逻辑类)。

6 UI 层 (用户界面) 的设计及对中间层的调用

6.1 添加网站

在解决方案上单击右键，执行 "添加-->新建网站-->ASP.NET 网站"，命名为 "新闻发布系统"。

6.2 添加动态网页

在网站中添加所需的 Web 窗体，生成有代码隐藏页的.aspx 文件，首页等前台页面直接放在网站根目录下，而后台管理相关网页，则放置于根目录下的 admin 文件夹中，以便于管理，项目结构如图 3 所示。以下列出部分代码页 (.cs) 文件中对中间层的调用代码：

(1) 根据选择的新闻 ID 取出该条新闻的具体代码：

```
//定义了 News 对象,因此在代码文件头部必须引用 Model
  using Model;
//在用户界面(UI)的隐藏代码中,都是靠创建 BLL 层对象来与
```

```
//数据库进行交互的,必须引用 BLL
  using BLL;
  //对象 n 必须定义为 public,否则.aspx 页面无法访问
    public News n = new News();
    //页面加载事件
   protected void Page_Load(object sender, EventArgs e)
   {
      string newsId;
 newsId = Request.QueryString["newsId"].ToString().Trim();
    // 构造 BLL 逻辑层的 NewsManage 对象 nm,实现对数据
//库的操作
      NewsManage nm = new NewsManage();
      //调用 nm 对象的 SelectById 逻辑方法,返回值为 News
//类的查找记录,赋给
//News 对象 n
    n=nm.SelectById(newsId);
     }
```

（2）后台登录代码

```
  //登录按钮单击事件
protected void LoginButton_Click(object sender, EventArgs e)
   {
     //创建注册用户对象
  Users rs = new Users();
  //通过对象的属性(set)给对象的字段赋值
     rs.Uname = Login1.UserName;
     rs.Pwd = Login1.Password;
      //创建注册用户管理对象
     UsersManage um = new UsersManage();
      //如果用户名和对于密码存在于数据库中
     if (um.IsUserPwd(rs))
     {
     //   登录成功,创建 Session 变量
        Session["username"] = Login1.UserName;
        Response.Redirect("manage.aspx");
     }
     else
       //验证失败,弹出提示对话框
     {
            Page.ClientScript.RegisterStartupScript   (Page.
GetType (), "message", "<script language='javascript' defer>
alert('用户名或密码错误,请重试!');</script>");
     }
  }
```

（3）用户注册代码

```
   //注册按钮单击事件
  protected void RegisterButton_Click (object sender,
EventArgs e)
  {
     bool flag = false;
     Users rs1 = new Users();
     rs1.Uname = uName.Text;
     UsersManage um1 = new UsersManage();
     if (pwd.Text ! = pwd1.Text)
     {
          Page.ClientScript.RegisterStartupScript   (Page.
GetType (), "message", "<script language='javascript' defer>
alert('两次密码不一致!');</script>");
     }
//IsUser 方法用来判断要注册的用户名在数据库中是否已存在
     else if (um1.IsUser(rs1))
     {
          Page.ClientScript.RegisterStartupScript   (Page.
GetType (), "message", "<script language='javascript' defer>
alert('用户名已存在!');</script>");
     }
     else
     {
        Users rs = new Users();
        rs.Uname = uName.Text;
        rs.Pwd = pwd.Text;
        rs.Email = email.Text;
         rs.Uauthority  =  Convert.ToInt32 (DropDownList1.
SelectedValue);
        UsersManage um = new UsersManage();
        flag=um.AddUser(rs);
        if (flag)
        {
            Page.ClientScript.RegisterStartupScript   (Page.
GetType (), "message", "<script language='javascript' defer>
alert('用户注册成功!');</script>");
          //注册成功后延迟 5 秒自动返回登录界面
    Response.Write("5 秒后自动返回登录界面,请稍后...");
          string strRedirectPage = "login.aspx";
          string strRedirectTime = "5";
          string strRedirect = string.Format ("{0};url={1}",
strRedirectTime, strRedirectPage);
          Response.AddHeader("refresh", strRedirect);
        }
        else
        {
            Page.ClientScript.RegisterStartupScript   (Page.
GetType (), "message", "<script language='javascript' defer>
alert('注册失败!');</script>");
        }
     }
  }
```

（4）更新新闻代码

```
 //页面加载事件,显示当前新闻各字段取值
 protected void Page_Load(object sender, EventArgs e)
 {
     News n = new News();
     NewsManage nm = new NewsManage();
     n = nm.SelectById(Request["newsId"]);
```

```
    if (! IsPostBack) //首次载入页面时绑定所选新闻内容和
//页面控件
        {
        DataTable dt = new DataTable();
        CategoryManage cm = new CategoryManage();
        dt = cm.SelectAll();
      //DropDownList1 为显示新闻分类的下拉列表框控件
        DropDownList1.DataSource = dt;
        DropDownList1.DataTextField = "name";
        DropDownList1.DataValueField = "id";
        DropDownList1.DataBind();
        DropDownList1.SelectedValue = n.CaId;
        txtTitle.Text = n.Title;
       // content1 为 textarea 控件的 id 值
        content1.Value = n.Content;
      }
  }
//更新按钮单击事件
   protected void NewsUpdateButton_Click (object sender,
EventArgs e)
  {
    News n = new News();
    n.Id = Request["newsId"];
    n.Title = txtTitle.Text;
    n.CaId = DropDownList1.SelectedValue;
    n.Content =content1.Value;
    NewsManage nm = new NewsManage();
    if (nm.Update(n))
    {
      Response.Write("修改成功！");
    }
    else
    {
      Response.Write("修改失败！");
    }
   }
```

(5) 删除新闻代码

```
    NewsManage nm = new NewsManage();
        //Delete 方法返回值为 bool 型的 true 或 false
    if (nm.Delete(Request["newsId"]))
    {
      Response.Write("删除成功！");
    }
    else
    {
      Response.Write("删除失败！");
    }
```

7 结语

(1) 三层架构层间关系。

UI 层向上直接与用户进行交互，向下则通过访问 BLL 层(业务逻辑层) 实现具体业务，而 BLL 层只是逻辑上的功能定义，具体功能是靠 BLL 层向下访问 DAL (数据访问层) 来实现的，即只有 DAL 层是直接访问数据库的一层。而 Model 层(实体层) 则为各个层实现以对象的方式表现数据提供了支持。

(2) 数据库的删除触发器。

在系统中删除新闻和删除新闻类别时都会产生级联式删除，为此采用了 instead of delete 触发器 (trigger)，如：要删除某条新闻，先要删除对这条新闻的评论；同样，删除新闻类别，要先删除此类别新闻的所有评论，然后再删除所有此类新闻，最后再删除此类别。本系统开始分别添加了两个 instead of 触发器，来代替新闻和新闻类别的 delete 操作，后来发现删除类别时，对此类别的新闻删除不彻底，分析是由于两个 instead of delete 触发器产生了嵌套：删除类别触发器(trigCategoryDelete) 中嵌套了删除新闻触发器 (trigNewsDelete)，最后通过修改被嵌套的删除新闻触发器 (trigNewsDelete) 解决。关键代码如下：

1)修改前：delete from news where id=@newsId

2)修改后：delete from news where id in (select id from deleted)

(3) 数据库中字段取值是否为空的判断。

判断数据库 News 表的记录中 content 字段值是否为空，正确语句应为：if (dt.Rows [0] [" content"] == DBNull.Value)，初学者很容易错写成：if (dt.Rows [0] [" content"] == null)，错误原因：dt.Rows [0] [" content"] 得到的是记录中的 content 字段值，是值类型；而 null 属于引用类型，两者不会相等，比较结果永远为 false。

图 3 解决方案资源管理器中的项目

参考文献

[1] 王翔. ASP.NET 4.0 网站建设基础教程. 北京：北京邮电大学出版社，2012.

[2] 马骏. C# 程序设计及应用教程 2 版. 北京：人民邮电出版社，2009.

(收稿日期：2012-10-10)