

基于 ASP.net 的三层结构实现方法研究^{*})

范振钧

(通化师范学院计算机科学系 吉林通化 134002)

摘要 本文论述了三层结构基本原理及其发展历程、Asp. Net 的特性以及 Asp. Net 系统组成结构,给出了 .net 平台上基于组件方式的三层结构的实现方法,并通过一个在线考试系统登录模块的实现,讲述了该方法在实际的软件开发中的实现过程。

关键词 B/S, ASP. Net, 组件, C#

Implementation Method of ASP.net-based the Three Layers

FAN Zhen-Jun

(Department of Computer Science, Tonghua Normal University, Jilin Tonghua 134002)

Abstract The basic principle and development of three layers, the characteristic and structure of Asp Net are discussed. At the same time, the implementation method of three layers based on component is emphasized by the example of login module using C# language.

Keywords B/S, ASP. NET, Component, C#

1 传统两层结构

在过去的系统开发过程中,Client/Server 体系结构得到了广泛的应用,其特点是:应用程序逻辑通常分布在客户端和服务端两端,客户端发出数据资源访问请求,服务器端将结果返回客户端。但 Client/Server 结构存在很多体系结构上的问题,比如:当客户端数目激增时,服务器端的性能会因为负载过重而大大衰减;一旦应用的需求发生变化,客户端和服务端的应用程序都需要进行修改,给应用维护和升级带来了极大的不便;大量的数据传输增加了网络的负载等等。因此,目前数据库应用程序的开发已经从传统的 C/S 结构向三层结构转变。

2 三层结构介绍

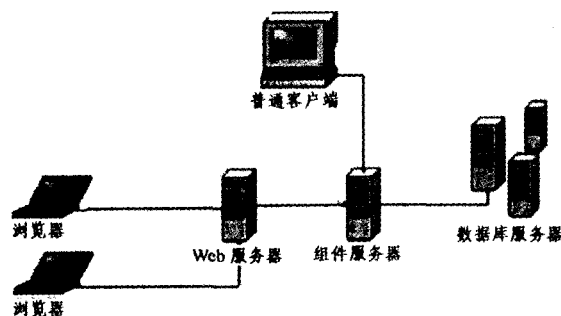


图 1 三层结构配置图

所谓三层体系结构,是在客户端与数据库之间加入了一个“中间层”,也叫组件层。这里所说的三层体系,不是指物理上的三层,不是简单地放置三台机器就是三层体系结构,也不仅仅有 B/S 应用才是三层体系结构。三层是指逻辑上的三层,即使这三个层放置到一台机器上。其基本物理配置图如图 1 所示。

三层体系的应用程序将业务规则、数据访问、合法性校验

等工作放在中间层进行处理。通常情况下,客户端不直接与数据库进行交互,而是通过 COM/DCOM 通讯与中间层建立连接,再经由中间层与数据库进行交互。三层结构原理如图 2 所示。

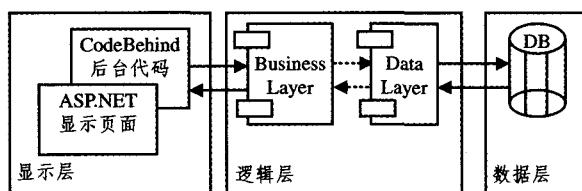


图 2 三层结构原理图

3 用 ASP.NET 部署三层架构

3.1 ASP.NET 简介

ASP.NET 是微软公司推出的一种 Internet 编程技术,它采用效率较高的、面向对象的方法来创建动态 Web 应用程序。在原来的 ASP 技术中,服务器端代码和客户端 HTML 混合在一起,常常导致页面的代码冗长而复杂,程序的逻辑难以理解。ASP.NET 是一种独立于浏览器的编程模型,可以在最新版本的 IE、NetscapeNavigator 等被广泛使用的浏览器上运行。ASP.NET 是一种建立在通用语言上的程序构架,是一个已编译的、基于 .NET 的环境,把基于通用语言的程序在服务器上运行。程序在服务器端首次运行时进行编译,比 ASP 即时解释程序速度上要快很多。微软公司发布了 4 种与 .NET 兼容的语言(包括 Visual Basic .NET、C#.NET、Visual C++.NET 和 JScript .NET)创作应用程序。

3.2 ASP.NET 特点

(1) 适时更新

管理员不必关掉网络服务器或者甚至不用停止应用程序的运行就可以更新应用文件。应用程序文件永远不会被加锁,因此甚至在程序运行时文件就可以被覆盖。当文件更新

^{*}) 吉林省教育科学“十五”规划课题(项目编号: B415134)。范振钧 讲师,硕士,主要研究方向:网络与数据库。

后,系统会温和地转换到新的版本。

(2)ASP.NET 采取“code-behind”方式编写代码

使得代码更易于编写,结构更清晰,降低了系统的开发与维护的复杂度和费用。

(3)基于 ASP.NET 技术的系统结构模型

ASP.NET 结构天然就是一个三层系统: UI 层、业务逻辑层和数据层。ASP.NET 系统结构如图 3 所示。

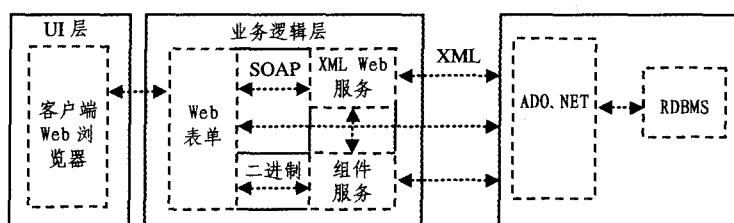


图 3 Asp.net 系统结构图

其中 UI 层负责与用户交互,接收用户的输入并将服务器端传来的数据呈现给客户。业务逻辑层负责接收浏览器传来的请求并将请求传给数据层,同时将请求处理结果发给浏览器。它由 Web 表单、XML Web 服务和组件服务组成。其中 Web 表单是 ASP.NET 应用程序的核心所在,是向客户呈现数据和信息的基础,也是响应和处理客户与显示的 Web 表单交互生成的信息和数据的基础。在本文给出的方法中,我们可以使用任何页面编辑工具如 dreamwaver、frontpage 等编辑 Web 表单。本层不负责任何业务逻辑的处理,只是负责通过页面控件向 Business-Layer 层传递参数,然后根据 Business-Layer 层的处理结果所传回来的参数,改变页面的各种显示方式,呈现给用户。UI 层的控制逻辑在 .NET 中是通过“code-behind”方式以 .aspx.cs 文件存放的。

业务逻辑层负责所有的业务逻辑的处理,在本文中它负责接受 UI 层传过来的参数。根据参数,确定自己的业务规则,然后为了程序设计的实用方便在逻辑层中添加了一个更低层组件 sqldb,负责与数据库相关的存储操作。业务逻辑层接受参数、确定业务规则后,直接调用 sqldb 组件,处理来自 UI 层的请求,把处理结果返回给 UI 层。业务逻辑层在 .net 中是以类库或 Web service 形式表现的。

数据层是通过 ADO.NET 操纵数据为事务逻辑层提供数据服务,如存储数据操作结果、返回数据检索结果等。在数据层中,为了改进应用程序的性能,可以引入存储过程。

由于 .net 结构本身的特点决定了在 .net 平台下,实现基于组件的三层结构方法变得非常简便、快捷。

4 实现方法

下面以一个登录模块为例子,采用 C# 程序设计语言,分别说明各层的实现。

4.1 数据层的实现

我们首先在 Sqlserver 2000 数据库中建立一个数据库 lwexam,在 lwexam 中建表 userinfo,表的结构如图 4 所示。

4.2 业务逻辑层的实现

(1)在 ms_vs2005 中新建网站,网站名称为 login,系统会出现名为 login 的解决方案。

(2)选定 login,点击右键,选择添加-新建项目->类型选为类库,名字 dbSql。这样就新建了一个组件 dbsql,它负责最底层 sql server 数据库的存取。在该组件的类中写入如下代码:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
namespace sqldb
```

```
{
    public class sqldb
    {
        public SqlDataReader reader(string xx)
        {
            SqlConnection conn=new SqlConnection("server=rj2;
            database=lwexam;uid=sa;pwd=jlthfjq");
            conn.Open();
            SqlCommand comm=new SqlCommand("select userid
            from userinfo where userid='"+xx+"'",conn);
            SqlDataReader read=comm.ExecuteReader();
            return read;
        }
    }
}
```

该代码的功能是:自定义一个方法 reader,接受上一层传过来的参数 xx,然后在 SQL server 数据库中的表 userinfo 中查询名字为变量 xx 中的信息,并返回一个 DataReader,数据集合。从代码中可以看出该组件主要功能是为方便数据存储,封装了所有关于数据库存储操作的底层信息,方便了上层业务逻辑的实现。该组件对应图 2 中的 Data-Layer 层。

列:

键	ID	名称	数据类型	大小	空	默认值
	☒	id	int	4	☐	
		userid	char	10	☒	
		username	char	10	☒	
		password	char	10	☒	
		power	int	4	☒	
		email	char	30	☒	

图 4 userinfo 表的结构

(3)选定 login,点击右键,选择添加-新建项目->类型选为类库,名字 logic。由于该组件通过 sqldb 组件实现数据库存取操作,因此点击该组件->添加引用->项目->sqldb,然后在该组件的类中写入如下代码:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
namespace sqllogic
{
    public class sqllogic()
    {
        private string fanhui;
        public string xianshi(string yy)
        {
            sqldb sqldb mm=new sqldb.sqldb();
            SqlDataReader nn;
            nn=mm.reader(yy);
            if (nn.Read())
```

```

    {
        fanhui=nn["userid"]. ToString();
    }
    else
    {
        return "该用户不存在,请重新登录";
    }
}
}
}

```

从上述代码可以看出,组件 logic 负责登录逻辑的处理,比如判断用户是否合法以及合法性判断后相应的处理逻辑。该组件对应图 2 中的 Bussiness-Layer,它通过向低层组件 dbsql 发送各种数据库存储要求,实现业务逻辑的所要求的具体的数据库存取操作。

理论上 sqldb 和 logic 两个组件在任何机器上单独编译成. dll 文件即可直接使用,充分实现了代码分离。

4.3 表示层的实现

新创建网站时系统会有一个默认的页面文件 default. aspx。在该页面文件中,通过放置一些 textbox 文本框,button 按钮,lable 控件实现登录表单。在后台 default. aspx. cs 中写入如下代码:

```
protected void Button1_Click(object sender, EventArgs e)
```

```

    {
        sqllogic. sqllogic xx=new sqllogic. sqllogic();
        Label1. Text=xx. xianshi(this. TextBox1. Text);
    }
}

```

从上述代码可以看出,表示层只是调用组件 logic 提供的方法 xianshi(string xx),通过页面控件传递参数。具体的逻辑处理完全由逻辑层的组件负责。理论上,表示层可以通过各种页面编辑工具制作,逻辑层可以在任何机器上存在,二者互不干扰,表示层的修改不影响逻辑层,反之亦然。这就充分方便了代码的复用以及应用程序的扩展。

小结 在 ASP. NET 中实现三层结构还有很多其他的方法,本文只是通过一个例子,对基于组件的开发方法做了一个简要的探讨。Asp. Net 天然的结构特性决定了它在 B/S 结构的软件开发中必将占有重要的地位。

参 考 文 献

- 1 Chris ULLman Chris Goode Asp. Net 入门经典. 北京:清华大学出版社,2002
- 2 廖信彦. Asp. NET 技术参考. 北京:中国铁道出版社,2001
- 3 石志国. Asp. Net 程序设计实用教程. 北京:电子工业出版社,2006
- 4 启明工作室. Asp. Net 网络应用系统开发与实例. 2005

(上接第 281 页)

表 1 各风险缓解策略对概率矩阵 P 和影响矩阵 M 的影响情况

风险缓解策略序号	影响类型	影响编号	工作分解元素	风险项序号											
				1	2	3	4	5	6	7	8	9	10	11	12
1	P	1a	1.2.1	0.60											
	P	1b	1.2.2								1.50				
	P	1c	1.2.3			0.80						0.80			
	P	1d	1.2.2.3				0.70								
	P	1f	1.2.6.2												0.80
	M	1g	1.2.3							20					
2	P	2a	1.2.1	0.20											
	P	2b	1.2.1.1							0.80					
	P	2c	1.2.2								1.50				
	P	2d	1.2.2.3				0.60								
3	P	3a	1.2.3									0.20			
	P	3b	1.2.4										0.30		
4	P	4a	1.2.2								2.00				
	P	4b	1.2.2.1			0.60									
	P	4c	1.2.2.3		0.70										
	P	4d	1.2.2.3				0.90								
	M	4e	1.2.3.2						20						
5	P	5a	1.2.1	0.50											
	P	5b	1.2.2.3				0.90								
6	P	6a	1.2.1	0.60											
	P	6b	1.2.2.1			0.90									
	P	6c	1.2.2.3				0.90								
7	P	7a	1.2.3									1.20			
	P	7b	1.2.4										1.10		
8	P	8a	1.2.3									0.80			
	P	8b	1.2.3.1					0.8							
9	P	9a	1.2.2								1.20				
	P	9b	1.2.2.1			0.90									
10	P	10a	1.2.4										0.80		
	P	10b	1.2.6.1											0.90	
11	P	11a	1.2.3									0.90			
	M	11b	1.2.4									20			
	M	11c	1.2.6.3										25		
	M	11d	1.2.6.1											10	
12	P	12a	1.2.6.2												0.30
13	M	13a	1.2.6.3												5