

基于 .NET 平台的三层架构软件 框架的设计与实现

高 扬

(贵州大学 计算机科学与信息学院, 贵州 贵阳 550025)

摘 要:面向对象和构件的软件开发平台,几乎是现在信息系统开发的必备之物,通过在平台之上融入三层架构及复用技术来构建一个功能丰富,结构灵活的软件框架也成为现在应用系统开发的主流思想。在介绍分析软件复用技术和三层架构思想的基础之上,研究并提出了基于 .NET 环境的三层架构信息系统软件框架设计的基本思路和方法,同时运用 C#语言实现了该软件框架的代码自动生成器。信息系统开发时通过运用该软件框架代码生成器可以很方便的生成一个 MIS 雏形,从而能有效的提高信息系统软件的开发效率和软件标准化及构件化水平。

关键词:软件复用;三层架构;软件框架

中图分类号:TP315

文献标识码:A

文章编号:1673-629X(2011)02-0077-04

Design and Realization of Three Layers Structure Software Framework Based on .NET

GAO Yang

(College of Computer Science and Information, Guizhou University,
Guiyang 550025, China)

Abstract: The object-oriented and components software development platform is almost essential things in information systems development. Three layers structure and software reuse technology are integrated in platform to construct a feature-rich and flexible software framework. It is popular in application system development. A software framework for information systems, which is based on the ideas of the software reuse and three layers structure, was designed and implemented in C# under the .NET platform. It can make it easy to generate a MIS prototype when developing an information system, finally improve the efficiency of information systems software development and the level of software standardization.

Key words: software reuse; three layers structure; software framework

1 软件框架

随着软件复用技术和软件构件化思想的深入研究,现代管理信息系统的设计也不再是以前的“程序=数据结构+算法”,而更多的是对软件框架和业务流程的综合研究与设计。软件框架是 MIS 中一组抽象构件以及构件间的接口所组成,在软件开发中开发人员可以根据实际需求对框架进行对象实例化与代码重构,从而快速形成一个“半成品”的应用系统程序,然后通过对业务流程的分析与设计从而完成整个应用系统的设计与实现。在一个面向对象的信息系统中,整

个系统的功能是由各对象之间的相互协作来完成,虽然这些对象之间的具体交互是由企业的实际业务流程所决定的,但可以从系统中提取出软件的框架结构,对 MIS 进行分层框架设计,实现软件最大粒度的可重用性。MIS 项目开发中一个面向对象的、可复用的信息系统软件框架已经成为一个软件开发团队的核心物质资本,而且更重要的是信息系统软件框架将是一个新系统的开发周期、开发成本及软件维护的核心影响因子^[1-4]。任何一个 MIS 软件框架都要有其支持的开发平台,Microsoft .NET 是 Microsoft 支持下一代 Internet 软件和服务的平台,作为新一代的开发平台,它彻底支持面向对象技术,适于快速的构建灵活的、功能强大的 MIS 软件。

文中将结合 .NET 平台所提供的相关技术,构建出一个具三层架构设计思想、面向对象、可复用的信息系统软件开发框架。

收稿日期:2010-06-04;修回日期:2010-09-18

基金项目:贵州省贵阳市科技型中小企业技术创新基金(市科技局〔2009〕筑科成合同第 5-013 号))

作者简介:高 扬(1984-),男,湖北安陆人,硕士研究生,主要研究方向为数据库技术。

2 三层架构思想

在管理学中有一个重要的概念即企业组织结构,企业组织结构是分层思想在企业中的重要应用,企业组织结构的目的是以求有效合理的把企业各层成员组织起来,为实现企业运作和发展目标而相互协同努力。在软件框架的设计时,分层结构是最常见也是最重要的一种结构,虽然软件框架分层的目的和形式跟企业分层有所不同,但都有一个共同目标:以求有效合理的组织相关构件,使其更高效的完成协同任务。在分层软件框架设计时最流行的是三层架构设计,任何一个系统从应用逻辑上对其进行抽象细分,均可划分为三层,自下至上分别为:数据访问层(DAL层)、业务逻辑层(BLL层)和表示层^[5-7]。在软件开发设计时还会用到一些通用辅助类和方法,如数据库访问类、事务处理类等,为了实现各个模块之间的相互复用,在本次软件架构设计时也将其分离出来,作为一个独立模块。在企业信息系统软件中整个系统操作的对象就是数据库中的数据表、视图等,为了便于在各层中相互传递,在设计时也将数据对象的实体和方法进行分离,将其抽象为一个共用实体类模块。根据以上设计思想,整个软件框架的三层框架形式如图1所示:

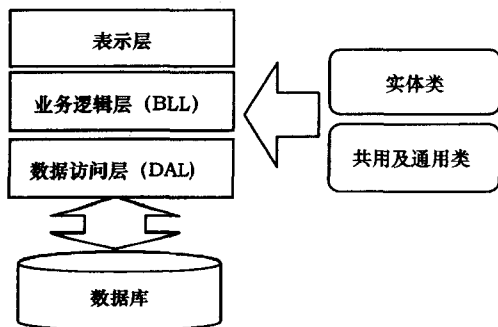


图1 三层架构

3 软件框架的设计与实现

3.1 数据访问层的设计

数据访问层(DAL层):用于实现信息系统对数据库的操作,完成业务流程对数据库中数据的插入、更新等操作。在管理信息系统中用户操作相关界面完成对应的业务流程的操作,但无论是什么业务流程最终反映到软件系统中则是对数据库中相关数据表单的数据进行操作,所业在软件框架中可以将数据访问进行深入抽象,将其分为数据库的查询运算、插入运算、修改运算及删除运算。这样对应的每个业务流程只需要指定相关的数据表或视图,就可以根据表中的数据项自动生成相关数据操作^[8-10]。在开发中选择目前使用最为广泛的关系数据库系统SQL2005,为了避免代码里出现很多SqlConnection、SqlDataReader等类和方

法,同时也为了让代码的编写及后期维护更加清晰简单,因此框架设计时采用了微软所提供的一个静态类SqlHelper。SqlHelper类通过一组静态的重载方法来封装数据访问功能,开发时可以通过调用SqlHelper的静态方法来对数据库的进行交互运算。除了公共方法外,SqlHelper类还包含一些专用函数,用于管理参数和准备要执行的命令,不管客户端最终采用什么样的方法调用实现,所有命令都将通过SqlCommand对象来执行。

在SqlHelper中ExecuteNonQuery方法用于执行不返回任何行或值的命令,通过设置命令参数可能用于执行数据库更新,也可用于返回存储过程的输出参数。ExecuteReader方法用于返回SqlDataReader对象,该对象包含由某一命令返回的结果集。ExecuteDataset此方法返回DataSet对象,该对象包含由某一命令返回的结果集。ExecuteScalar此方法返回一个值,该值始终是该命令返回的第一行的第一列。因为所有的命令均都是通过SqlCommand对象来执行,而SqlCommand能够被执行之前所有参数都必须添加到Parameters集合中,并且必须正确设置Connection、CommandType、CommandText和Transaction参数。数据查询运算首先根据所选择的实体类创建查询字符串strSQL,然后调用ExecuteReader方法实现数据查询,方法会将查询的List实体类对象结果值返回。数据的插入运算时可以运用ExecuteScalar方法配置相关参数来实现插入运算,通过方法返回插入后对应的ID号从而可以实现主从表的插入。数据的修改及删除运算可以通过ExecuteNonQuery方法来完成。下面将以数据表Storage_StockOutType为例,通过查询操作来展示软件框架代码的具体实现。

```
public List<Storage_StockOutTypeInfo> GetStorage_StockOutType(Storage_StockOutTypeInfo entity) //通过传递实体类对象来完成查询
{
    StringBuilder strSQL = new StringBuilder();
    //创建相关调用参数
    strSQL.Append("SELECT * FROM Storage_StockOutType WHERE 1=1");
    if (entity.ID != 0)
    {
        strSQL.Append(" and ID=" + entity.ID + "");
    }
    ..... //通过实体对象拼接查询字符串 strSQL
    List<Storage_StockOutTypeInfo> objItems = new List<Storage_StockOutTypeInfo>(); //定义实体类的 List 返回像
    using (SqlDataReader rdr = SqlHelper.ExecuteReader(SqlHelper.ConnString, CommandType.Text, strSQL.ToString(), null)) //调用 ExecuteReader 方法完成相关
```

操作

```

{ while ( rdr. Read() ) //调用 Read 方法来循环读取数据
{
    Storage_StockOutTypeInfo objInfo = new Storage_StockOutTypeInfo();
    if ( rdr[ "ID" ] != DBNull. Value ) //根据数据库表字段来判断并构建反回的 List
    {
        objInfo. ID = Convert. ToInt32( rdr[ "ID" ] );
        .....
        objItems. Add( objInfo ); //调用 Add 方法添加符合条件的实体对象到 List 中去
    }
}

```

在整个数据访问层的设计时关键是要正确合理的设计相关方法的处理参数,另外处理数据库时需要保证数据库的完整性,因而在数据处理时必须同时通过事务处理来实现,通过对 SqlHelper 的深入研究发现在事务处理时处于同一事务处理中的各静态类中必须使用同一 connection,为了体现三层框架的设计思想,定义并实现了一个 Connection 静态创建类,通过调用相关方法可以创建 connection 参数。

数据层的处理流程如图2所示:

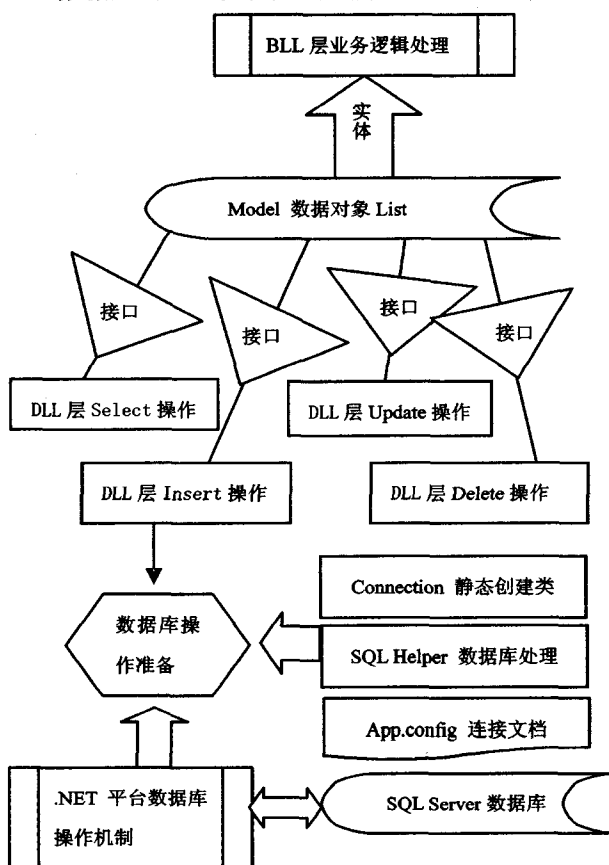


图2 数据层处理流程

3.2 业务逻辑层的设计

业务逻辑层(BLL层):用于实现数据业务流程,

该部分研究与设计在管理信息系统软件的开发过程中是系统实现的核心环节,用于对上下层之间的交互数据进行逻辑处理,实现对应的业务目标。在软件框架设计时因业务逻辑要根据具体的业务流程来决定,所以在软件框架中该部分的设计主要是设计通用的业务接口,通过这些接口访问数据访问层,从而完成相关业务操作。当有独特的业务流程时可以对 BLL 层中相关类进行继承,然后重载相关操作。可以发现不管是什么对象和业务,只需要选择相关的数据表或视图,然后根据表或视图的数据字段就可以用相同的开发思想自动的生成不同表的业务层通用操作代码。业务层主要操作实例如下:

```

public class Purchase_ApplicationBLL

```

```

{
    public static<返回参数> 业务功能1()

```

```

    { // 业务1实现代码 }

```

```

    .....

```

```

    public static<返回参数> 业务功能n() //当需要多层业务时,通过 TransactionScope 实现事务处理

```

```

    { // 业务n实现代码 }

```

```

}

```

3.3 界面层及实体类的设计

界面层:主要实现人机交互和软件展示,负责提供一个完美的业务操作界面供用户操作相关业务,同时对用户业务操作结果进行展示^[11,12]。在界面层的设计时可以根据用户所选的不同控件,通过 GET 和 SET 方法来完成实体类属性的操作,同时将实体类的值绑定到对应控件的属性上,以完成数据的自动获得与显示。因此在界面层代码生成时,只需要先选择对应的数据库操作对象(多表时同时指定主从表及外键),然后给各数据字段指定相关显示控件,就可以运用前面的思想自动生成相关 GET 和 SET 操作代码。其次在信息系统软件的整个界面上通常会有打印、查询、增加、删除、修改、审核、弃审等业务操作菜单及首页,上一页,下一页,最后一页等功能显示菜单,业务操作菜单操作完全相同,只是具体的业务实现有所不同,所以在现软件框架中,可以将业务操作菜单的框架实现,把业务流程作为接口函数在业务层去根据实际情况来实现。对于功能显示菜单,因为其功能完全相同所以其功能代码可以在框架中完全实现。

实体类(Model):是从数据库中的表抽象出来的对象类。在数据库管理系统中存储和操作的是数据库表及视图,而在信息系统软件中操作的是对象实例,所以软件框架设计时需要把关系数据库中的表、视图进行对象实体化。在设计时将所选择的表、视图映射为实体类中的类对象,将表中的字段通过 GET 和 SET 方法将其映射为对象的属性,这样就可以把数据库和

MIS 业务对象进行有机的结合,从而也形成了软件框架的实体类。

3.4 软件框架实现效果

通过以上三层架构软件的各层分析与设计,将其制作成一个软件代码自动生成的小工具,整个软件框架代码生成工具的总体界面如图 3 所示,通过配置数据库连接字符串后,程序将打开对应的数据库并显示操作界面,从左边可以选择对应的数据库表及视图,在右边上面选择相应的软件处理层,在数据库表或视图选择要用的数据字段,点查看代码则可生成各层代码。

在整个信息系统中,三层框架代码是信息系统的骨架也是文中所研究与开发的核心内容,通过所开发的软件框架生成工具可以自动生成三层框架的 DLL 层,BLL 层,界面层及 MODEL 类代码。下面以 MODEL 层代码为例(见图 3),通过在总界面中选择“MODEL 层”然后选择相关数据库表并点“查看代码”,即可生成 MODEL 层代码后点保存按钮即可将 MODEL 代码,保存为相关 MODEL 层文件,然后将生成的文件添加

到解决方案的 MODEL 层即可。

4 结束语

将三层架构思想引入到 .NET 平台下信息系统软件框架的设计之中,运用 C# 语言实现了一个比较完善的三层软件框架程序。在信息系统软件开发时可以将实际业务需求和软件框架进行有机的结合,运用软件框架程序来自动化生成信息系统的三层架构代码。从而降低系统的耦合性,增加系统的构件化水平,使系统各层之间分工明确,基础框架模块与扩展模块之间功能定位明确。对软件系统来说这样不仅可以缩短开发周期,降低开发成本;同时也有利于信息系统软件走向模块标准化,使软件的测试与后期的维护和升级更加方。另外一方面,随着软件框架代码的自动生成,开发人员可以将精力从繁琐的代码编写中解脱出来,而把主要时间放在软件的业务功能流程设计之中,这样设计出来的系统将更加符合客户的实际需求,提高了软件开发的成功率,提升了整个信息系统的业务工作性能。

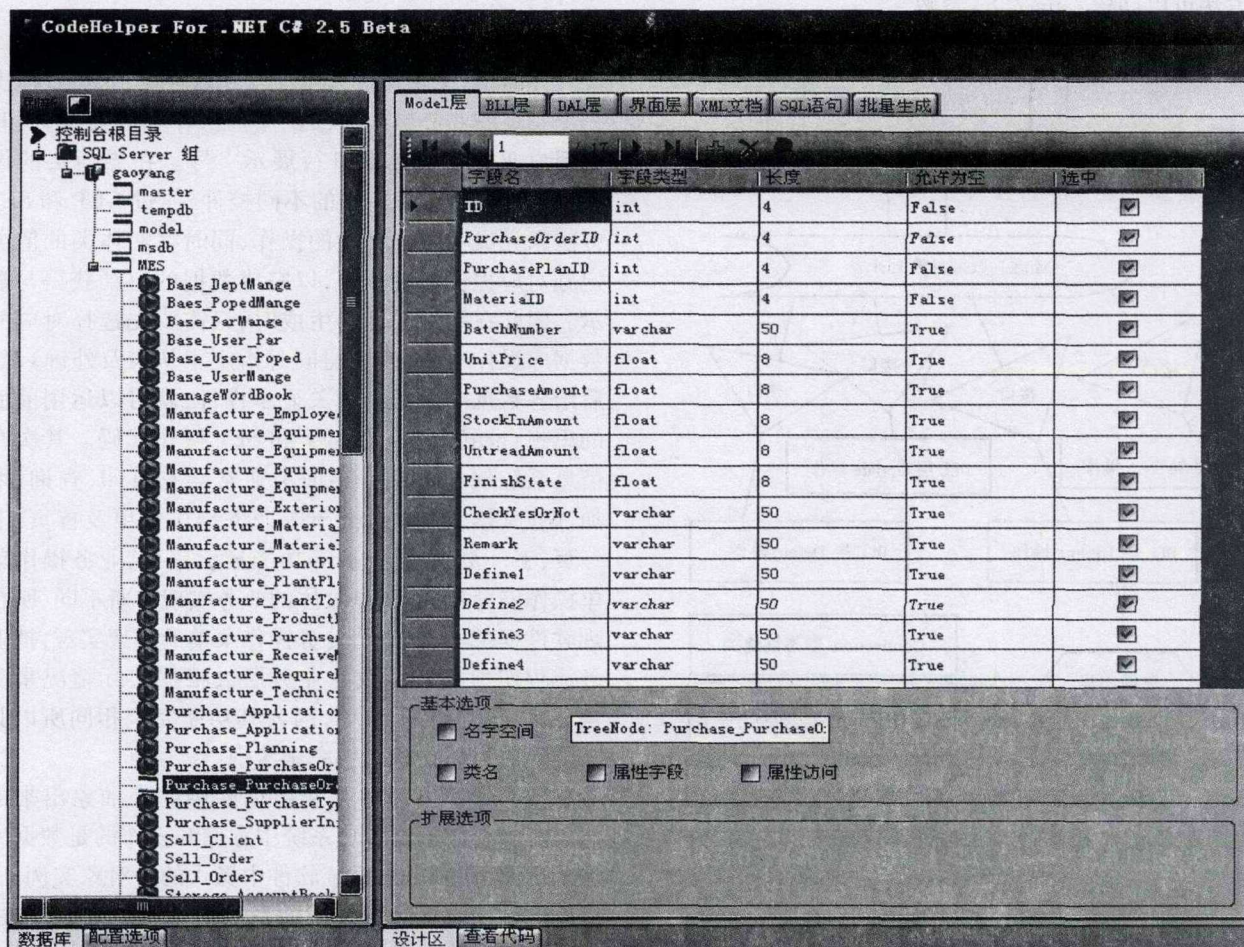


图 3 软件框架界面

这里 p 代表节点传输概率, N 为系统中节点总数, S 为平均吞吐量。将上式用时段为单位表示为

$$D = (m_1 + m_2) * (1 - \frac{1}{p} + \frac{N}{S}) \quad (11)$$

同理, RIBTMA 的时延可以表示为

$$D = (1 + m_2) * (1 - \frac{1}{p} + \frac{N}{S}) \quad (12)$$

MCBTMA 和 RIBTMA 的时延性能对比,如图 6 所示,可以看出由于在 Mesh 网络中引入了多信道,在同一时间可以允许多路数据流同时传输。随着系统中节点数的增多,MCBTMA 系统时延特性明显优于 RIBTMA。

5 结束语

在无线 Mesh 网络中多信道 MAC 协议与单信道 MAC 协议相比,设计上更加复杂。由于无线 Mesh 网络的多跳性,缺乏集中式控制、网络拓扑时变等特点,使多信道 MAC 协议的设计面临诸多困难。文中给出了一种带有 MIMO-OFDM 技术的收发机结构,通过新型多天线收发机来构建多信道。随后提出了适用于无线 Mesh 网络的多信道 MAC 方案 MCBTMA,通过引入多路忙信号控制信道,在 MAC 层进行信道调度分配,达到多信道情况下系统性能的改善。系统性能数值仿真结果表明,引入 MCBTMA 的无线 Mesh 网络在系统吞吐量和时延方面性能有显著提高。本方案中仅引入了 2 个控制信道和 2 个数据信道,系统模型较为简单,因此对系统性能改善有限。下一步工作中将继续研究多控制信道与多数据信道情况。

参考文献:

- [1] 姜永,郑明春,李国强. IEEE802.16 中的无线 Mesh 网络研究[J]. 计算机技术与发展,2008,18(5):234-237.

(上接第 80 页)

参考文献:

- [1] 甄镭. NET 与设计模式[M]. 北京:电子工业出版社,2005.
- [2] 程春蕊,刘万军. 高内聚低耦合软件架构的构建[J]. 计算机系统应用,2009,7:19-21.
- [3] Larus J R, Ravi R. Transactional Memory[M]. New York: Morgan and Claypool,2007.
- [4] 罗剑. 基于组件的 ERP 软件框架的设计与实现[J]. 计算机应用与软件,2006,23(9):78-80.
- [5] 郝雯,艾玲梅,王映辉. 三层结构软件框架扩展点实现方法[J]. 计算机应用,2009,29(9):2541-2543.
- [6] Hsiao Yuan-Du. A study on key failure factors for introducing enterprise resource planning[J]. Human Systems Manage-

- [2] 李玲娟,丁亮. 无线传感网中多跳路由算法的研究[J]. 计算机技术与发展,2010,20(6):55-58.
- [3] 司宏林,王晓蔚. 无线传感网多信道的 MAC 层协议[J]. 计算机技术与发展,2006,16(3):232-234.
- [4] HuangDi-Wei, Lin Phone, Gan Chai-Hien. Design and Performance Study for a Mobility Management Mechanism (WMM) Using Location Cache for Wireless Mesh Networks[J]. IEEE Transactions on Mobile Computing,2008,7(5):546-556.
- [5] Akyildiz F, Wang X, Wang W. Wireless mesh networks: a survey[J]. Computer Networks Journal,2005,5(9):23-29.
- [6] Zheng Jianliang, Ko Young-Bae, Shrestha D M. Emerging standards for wireless mesh technology[J]. IEEE Wireless Communications,2006,13(2):56-63.
- [7] Liu P, Tao Z, Lin Z. Cooperative Wire-less Communications: A Cross-Layer Approach[J]. IEEE Communications Magazine,2006(6):84-92.
- [8] Choudhury R R, Yang Xue. Using Directional Antennas for Medium Access Control in Ad Hoc Networks[J]. Proc. ACM MOBICOM,2002,2(9):23-28.
- [9] Wu C, Li V. Receiver-initiated Busy-tone Multiple Access in Packet Radio Networks[J]. ACM SIGCOMM Computer Communication Review,1987,17(5):336-342.
- [10] 刘晓明,贺璟,戴政兵,等. OFDM 中基于导频的整数倍频偏估计算法及实现[J]. 计算机技术与发展,2010,20(7):28-31.
- [11] Kyasanur P, Jungmin. Multichannel mesh networks: challenges and protocols[J]. IEEE Wireless Communications,2006,13(2):30-36.
- [12] Shen Dongxu, Li V O K. Performance Analysis for A Stabilized Multi-channel Slotted ALOHA Algorithm[C]//Proceedings of the 14th IEEE 2003 International Symposium on Personal. Indoor and Mobile Radio Communications. [s. l.]:[s. n.],2003:249-253.

ment,2007,26(2):140-151.

- [7] 胡丹,龚俭. 基于 .NET 的分布式网络管理软件框架[J]. 计算机技术与发展,2006,16(3):90-91.
- [8] 王晟. Visual C#. NET 数据库开发经典案例解析[M]. 北京:清华大学出版社,2005.
- [9] 莫勇腾. 深入浅出设计模式[M]. 北京:清华大学出版社,2006.
- [10] Lorenz P A. ASP.NET 2.0 Revealed[M]. New York: Apress,2004.
- [11] 王孝明,胡健,陆坤,等. 基于 .NET 平台可复用软件框架的设计与实现[J]. 计算机工程,2004,30(22):76-78.
- [12] 李园,陈世平. MVC 设计模式在 ASP.NET 平台中的应用[J]. 计算机工程与设计,2009,30(7):3180-3184.