AOP 在基于. NET Remoting 的分布式 ERP 系统中的应用

Application of AOP in Distributed ERP System based on . NET Remoting

孙文静　（南京审计学院计算机系　210029）
傅　涛　（南京理工大学计算机系　210094）

摘要：本文以江旭铸造有限公司为背景，研究了在. NET Remoting 架构上开发分布式 ERP 系统的过程中，通过 AOP 技术实现横切关注点的模块化思想及实现过程。系统运行表明，这一技术增强了系统的可扩展性、代码的高效性及可复用性。

关键词：AOP　横切关注点　动态代理　ERP　应用

随着经济全球化进程的加快和国际竞争趋势的日益明显，入世以来，越来越多的国内企业依靠实现 ERP（Enterprises Resources Planning，企业资源计划）来提高自身在国际市场上的竞争力。ERP 涵盖了包括供应链、生产制造、内部管理、过程控制、销售和客户关系在内的一系列企业业务范畴。本文研究的是为江旭铸造有限公司开发分布式 ERP 系统的过程中，在. NET Remoting 架构上通过 AOP（Aspect – Oriented Programming，面向方面程序设计）技术实现了所有业务对象的数据访问功能以及分布式系统基于动态代理技术的通信功能。实际表明，这一设计思想使系统更易维护和扩展，同时减小了代码的冗余度，增加了代码的可复用性。

1　AOP 技术简介

现在，大多数 ERP 软件项目的开发都选择 OOP（Object Oriented Programming，面向对象程序设计）的编程方式。确实 OOP 已经表明了它处理一般行为的能力，但是，OOP 只是将公共属性和行为都封装在根对象中，却不能很好地处理横越多个（经常是不相关的）模块的行为。相比之下，AOP 填补了这个空白。AOP 是一种解决横切业务关注点的技术，它试图模块化某个软件层中散布于各个类（不管其领域如何）的公共属性和行为，从而增加软件的模块性。AOP 是一种扩展性的技术，它不会代替 OOP，而是对 OOP 编程范型上的改进。

一个复杂的系统可看作是由多个关注点组合实现的。江旭铸造有限公司的 ERP 系统包括如下的一些关注点：业务逻辑、数据访问、日志和调度信息、网络通信、数据安全、有效性验证等等。每一个关注点都牵涉到系统的几个业务子模块，如数据访问关注点会影响到所有的业务对象。把数据访问关注点封装成一个模块化单元（称为“方面”），各个业务对象在需要访问数据时实现对该方面的调用，而不必在自己的业务模块中单独实现。解决关注点的这种技术叫做动态横切（dynamic crosscutting），在多个模块中出现的关注点叫做横切关注点（crosscutting concerns）。

本系统部署在应用服务器上的中间件容器和部署在客户机上的客户接口层就是按照 AOP 的设计范型来设计的，以下将具体介绍。

2　基于 AOP 的系统体系结构及工作流程

本系统的体系结构分为四层，分别是数据库（DataBase）、中间件（MiddleWare）、中间件容器（MiddleWareContainer）和客户端（Client），如图 1 所示。其中中间件由数据访问层（DataAccessLayer）和业务逻辑层（BizLogicLayer）组成。客户端由客户接口层（ClientInterface）和用户图形界面（GUI）组成。中间件容器利用 AOP 技术实现了整个中间件的事务和安全规范。

下面以一次入库操作为例，来说明系统的工作流程：

（1）客户通过 GUI 输入入库单信息，并将该信息封装成入库单实体类；

（2）客户接口层通过代理调用库存模块中间件上的“入库”方法，将参数序列化为 XML 流，通过 TCP 信道传输到中间件容器；

（3）中间件容器将 XML 流反序列化，重新打包成入库单实体类；

（4）将入库单实体类传入对应模块的业务逻辑层，施以商务规则，从中得到有用的信息，并将结果传给数据访问层；

（5）数据访问层对这些入库单实体类进行解包，从中分离出基础信息；

（6）数据访问层调用 SQLDbBase 中重载的一系列底层

数据库的访问方法，将这些方法动态插入到中间件容器所提供的基于事务的通用数据库访问逻辑中的动作链中。并用这些方法访问数据库，将参数传给数据库服务器；

（7）数据库服务器根据传入的存储过程名和对应的参数执行存储过程，并将结果返回给 SQLDbBase。

（8）中间件容器将库存模块中间件的处理结果经过序列化和反序列化传给客户端的代理，再由代理传给 GUI，完成一次入库操作。

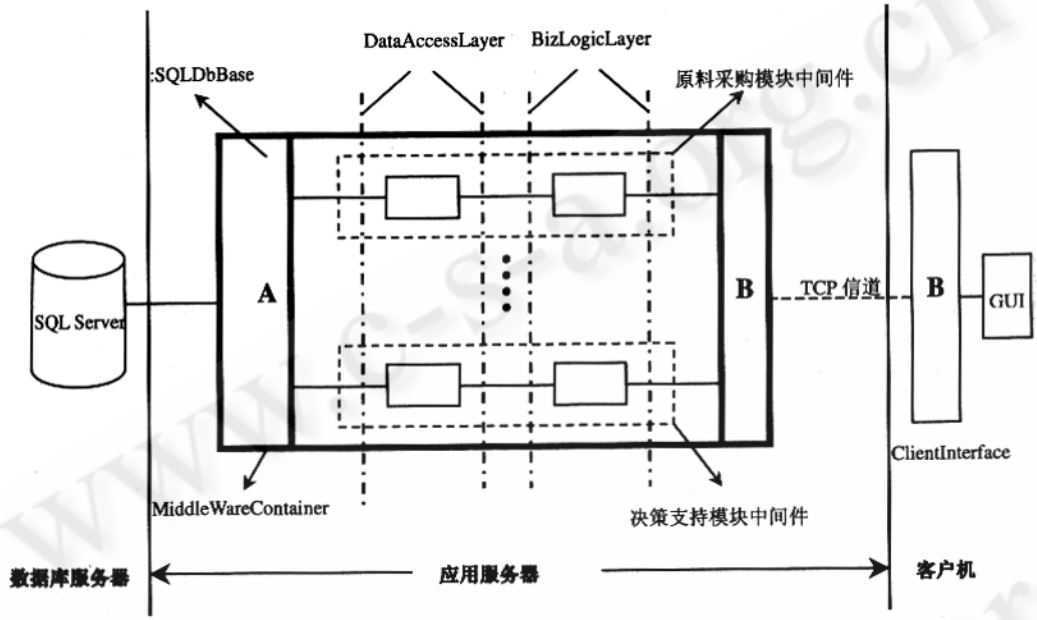

图 1　基于 AOP 的系统体系结构图

3　AOP 技术在系统中的实现

3.1　通过 AOP 技术实现各个模块对数据的访问

本系统按业务划分共包含了 10 大模块，具体是：原料采购模块、模具管理模块、设备管理模块、库存管理模块、人力资源模块、客户管理模块、财务管理模块、质量检测模块、市场销售模块和决策支持模块。

前面讲过，数据访问关注点会影响到所有的业务模块，是一个横切关注点。根据 AOP 的开发思想首先要独立地实现"数据访问"方面：即将所有的数据访问细节都封装在通用数据访问基类 SQLDbBase 中。然后，各个业务模块的数据访问层继承 SQLDbBase 类并调用基类中相应的数据库访问方法访问数据，而不需要单独实现数据访问细节，从而通过 AOP 技术实现了横切关键点的模块化。图 1 中 A 即为封装后的"数据访问"方面。

下面通过入库操作来具体说明 AOP 在实现库存管理模块的数据访问中的应用。

（1）首先定义通用数据库访问接口类 SQLDbBase。SQLDbBase 是各个模块的数据访问层的基类，它定义了对 SQLServer 数据库的通用访问方法。重要的对外接口定义如下：

private +BuildSqlCommand( ) //重载方法，构造一个新的 SqlCommand，配置与远程数据库的连接信息以及设置存储过程的参数。

protected +ExecStoredProc( ) //重载方法，负责执行存储过程

protected + ExecStoredProcInTrans ( ) //重载方法，支持基于事务的一组存储过程的执行

protected virtual void InnerDataHandler( ) //定义将要放在事务中的一系列对数据库的访问操作。

以下是实现 ExecStoredProcInTrans 方法的简略代码：

```
protected void ExecStoredProcInTrans( )
{
  SqlTransaction ObjTrans = BuildTransaction( ); //构造事务
  try
  {
    ObjTrans. BeginTrans( ); //启动事务
    InnerDataHandler( ); //数据库操作，在派生类中定义
    ObjTrans. Commit( ); //提交事务
  }
  catch( Exception E) //捕获异常
  {
    ObjTrans. Rollback( ); //回滚事务
  }
}
```

（2）定义库存管理模块的数据访问类 D_StockModule：SQLDbBase，它属于数据访问层。D_StockModule 是 SQLDbBase 的派生类，它实现了物品入库的三个动作（填写入库单、填写入库单明细、更新库存），并将他们定义在一个事务过程中。简略代码如下：

```
public int ExecProc_ItemInStock( ItemInStock ObjItemInStock, Item[ ]ObjItems)
{
  return base. ExecStoredProcInTrans( );//基于事务的数据库访问通用方法
}
```

```
override void InnerDataHandler( ref SqlCommand ObjSqlCommand) //需要包含在事务中的对入库环节的操作, //ObjSqlCommand 由 base. BuildSqlCommand( ) 构造
{
  base. ExecStoredProc ( " FillInStockVoucher " , ItemInStock. ObjItemInStock) ; //填写入库单, ItemInStock 是数据//库中入库单的实体类
  base. ExecStoredProc ( " FillInStockDetail " , Item [ ] ObjItems) ; //填写入库单明细
  base. ExecStoredProc( " UpdateStock" , Item[ ] ObjItems) ; //更新库存
}
```

3.2 通过 AOP 技术实现动态代理

传统意义上,进程通常被用作隔离的分界线,也就是说,在一个进程中运行的应用程序不能访问和破坏另一个进程中的内存。对于相互进行通信的应用程序来说,跨进程的通信是必要的。. NET 应用程序是在应用域中工作的,应用域可以看作是进程中的子进程。有了. NET,应用域就成为进程中新的安全分界线。不同的应用程序可以在同一进程中运行,但不能在同一个应用域中运行。在同一应用域中的对象可以直接进行交互,但是在访问不同应用域中的对象时,必须使用代理。. NET Remoting 可以用于在一个应用域中访问另一个应用域中的对象。不论两个对象是处于一个进程中,还是处于不同的进程中,或者甚至处于不同的系统中,都可以使用. NET Remoting。

在本系统中,. NET Remoting 用于通过网络进行通信的服务器和客户机,客户机调用服务器的方法时,使用了动态代理技术。因为每个业务模块都必须对中间件进行访问,所以应用 AOP 技术在应用服务器端设计了一个中间件容器(MiddleWareContainer),然后在客户端设计了统一的客户接口层(ClientInterfaceLayer),并在其中封装了创建动态代理以及和客户端进行通信的详细方法。

代理类工厂(ProxyFactory)主要负责在系统中配置客户端远程通道信息,生成中间层的动态代理,用以和应用服务器在基于 TCP 的信道上进行通信。图 1 中 B 即实现了基于 AOP 的动态代理模式。下面将具体说明它的实现过程。

(1) 首先定义 ProxyFactory 类,重要的对外接口及主要代码如下:

GetUri(ModuleType ObjModuleType) //获取注册的中间件的 Uri 地址。

+GetProxy(ModuleType ObjModuleType) //重载方法,适用于在不同信道上(HTTP/TCP)动态获取远程对象(中间件)的代理。

```
public static object GetProxy( ModuleType ObjModuleType)
{
Type[ ] ObjInterfaceType = GetInterfaceType( ) ; //获取中间件接口(MiddleWareInterface)定义的所有模//块接口的类型
  foreach ( Type ObjType in MiddleWareInterfaceType) //循环查找与 ObjType 相匹配的中间件接口中//的类型
  {
    if( ObjType. Type = = ObjModuleType)
    {
      object ObjProxy; //定义一个存放远程代理的空对象
      ObjProxy = WellKnownClientConfig ( ServerID, ServerPort, GetUri( ObjModuleType) , ObjType) ; //配置 TCP 信道,并获取远程对象的代理
      return ObjProxy;
    }
  }
}
```

(2) 方法类工厂(MethodFactory),主要负责通过反射动态执行需要调用的接口中的方法。接口定义如下,代码略。

DoProxyMethod(object ObjProxy, ModuleType TheModuleType, string MethodName, object[ ] InputParam) //将获得的远程代理作为参数,根据传入的方法名 MethodName 和参数 InputParam 动态执行代理中所定义的方法。

4 结束语

相对于 OOP 来说,AOP 的编程方式有效地实现了模块化横切关注点的实现,这为 ERP 开发提供了一个优秀的软件设计模式。本文完成的江旭铸造有限公司的管理信息系统在. NET Framework1. 1 + SQL Server2000 + Windows 2000 Advanced Server SP3 平台下开发,现已交付实际使用。实践显示,AOP 技术可大大简化程序代码,增强了系统的可扩展性。

参考文献

1 Simon Robinson, Burt Harvey, and so on. Professional C# (2nd Edition). Wrox Press Inc; ISBN:1861007043; 2002. 3.

2 Gregor Kiczales、John Lamping、Anurag Mendhekar、Chris Maeda、Cristina Videira Lopes、JeanMarcLoingtier 和 John Irwin, 1997, Aspect Oriented Programming, http://aspectj. org/documentation/papersAndSlides/ECOOP 1997 - AOP. pdf.