第16卷 第1期
2006年1月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 16 No. 1
Jan. 2006

.NET平台下ORM组件的研究与探索

洪 艺[1],胡华平[1],刘利枚[2]
(1. 国防科技大学 计算机学院,湖南 长沙 410073;
2. 湖南商学院,湖南 长沙 410205)

摘 要:目前 Object－Relationl Mapping(ORM)已经成为软件体系结构领域一个新的研究热点。文中在分析.NET平台下三种比较典型的ORM组件的基础上,结合正在进行的办公自动化系统开发的实际需要,提出并实现了能适合这一类系统使用的、基于.NET平台的轻量级ORM组件——MY ORM LAYER。最后描述了ORM组件未来发展的趋势。

关键词:对象关系映射;.NET平台;办公自动化
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1005－3751(2006)01－0013－04

Research and Implementation of ORM Component in .NET Platform

HONG Yi[1], HU Hua-ping[1], LIU Li-mei[2]
(1. School of Computer Science, National University of Defense Technology, Changsha 410073, China;
2. Hunan Business College, Changsha 410205, China)

Abstract: The Object－Relationl Mapping(ORM) is becoming a new focus of study. Under the analysis on three kinds of more typical ORM component of NET platform, combine an actual need of the office automated system, this paper proposes and realizes lightweight ORM package of NET platform—MY ORM LAYER, which is suitable for this kind of system to use. At last, described the development trend of ORM component in the future.

Key words: object relational mapping; .NET platform; OA

0 引 言

在.NET平台下,一个典型的三层系统是由表现层、业务层和数据层构成的。在业务层上大部分使用的是C#、VB.NET等面向对象的语言,以对象模型来处理复杂的业务逻辑;数据层采用的是SQL Server等建立在关系模型之上的数据库。由于对象和关系之间的"阻抗不匹配"问题,把面向对象的一些操作映射到关系数据库比较麻烦,需要写不少数据访问的代码,而这些代码总是重复的。经常需要修改业务层的代码和SQL语句来适应处在数据层的数据库表的变化,这使得系统难以维护。ORM是解决这个问题的有效方法。ORM全称Object Relational Mapping(对象关系映射),即在业务层和数据层中添加一个软件层,将面向对象编程所建立的对象在数据库中做一个映射,使之和数据库中的表建立一一对应的关系。把对表直接进行的操作,变成对类的属性和方法的操作[1]。这样大大降低了业务层和数据层的耦合度,提高了系统的扩展性和可维护性,提高了开发效率。在项目中使用ORM组件,能把开发人员从低级重复的劳动中解脱出来,使之有更多的时间关注于真实的商业需求上。

收稿日期:2005－04－12
作者简介:洪 艺(1980—),男,湖南长沙人,硕士研究生,研究方向为软件体系结构。

1 .NET平台下三种典型ORM组件介绍

.NET平台下的ORM组件实现的方式各有不同。但是决定一个ORM组件风格主要有4个方面:Mapping方式、关系映射方式、查询方法和建模方法。下面将主要从这4个方面对以下三种典型的ORM组件进行阐述。

1.1 ObjectSpaces

ObjectSpaces是微软计划在2006年正式发布的ORM组件。现在普遍使用的是.Net Framework 1.2 Alpha测试版里面带的ObjectSpaces,版本号1.2.30703.27。ObjectSpaces需要自己建立一个实体类,使用OSD、RSD、MSD三种XML文件来分别描述所有实体类的定义、关系数据库中所有表的信息、OSD和RSD之间的映射。ObjectSpaces可以支持非常丰富的Relations,OneToOne,ManyToMany,OneToMany等等。ObjectSpaces使用Microsoft创建的一套融入XPath和SQL思想的搜索语言——OPath语言进行查询。由于开发人员使用ObjectSpace时,必须先在数据库里建立好表结构,再根据表结构建立相关映射文件和实体类,因此它的建模方式是从DB→Class的。

1.2 NHibernate

NHibernate是基于Microsoft .NET Framework的OR－M持久框架,它从基于Java的Hibernate项目移植而

来。与 ObjectSpaces 一样，NHibernate 使用硬编码实体类来定义好类结构，但是只使用一种 XML 文件来描述数据库表结构和实体类结构的对应关系，以及实体类结构之间的 relations。NHibernate 也可以支持非常丰富的 Relations，包括 OneToOne、ManyToMany、OneToMany 等。NHibernate 查询用的是完全面向对象的语言——HQL 语言进行查询。与 ObjectSpace 一样，它的建模方式也是从 DB→Class 的。

1.3 XPO

XPO 是 DevExpress 公司的商业 ORM 产品。XPO 即 eXpress Persistent Objects for .NET，现在这里介绍的版本是 1.5。它是采用自定义属性的方式来实现 Class 与 Table 的映射，不像 ObjectSpace 用 XML 文档来保存映射信息，XPO 是用在类或者方法前面加特性(Attribute)来完成映射，是一种单纯的持久化类映射。它的映射可以支持：名称映射、索引建立、选择性映射、关系映射、实体类继承、级联更新、迟加载、锁定类型、NULL 处理等功能，基本上可以满足 CRUD 和类之间的关系维护。XPO 对 Class→DB 和 DB→Class 两种建模方式都支持，但是 XPO 偏重的是 Class→DB 的方式。该方式将根据业务系统中的类，通过类或方法本身的属性和特性自动生成数据库表结构自动生成对应的数据表，并建立表之间的关系。所以它并没有提供类代码生成器。

2 结合工程实际的考虑

2.1 办公自动化软件存在的实际问题

政府部门办公自动化系统是以公文处理和机关事务管理(尤其以领导办公)为核心，同时提供信息通讯与服务等重要功能，因此，典型的办公自动化应用包括公文管理、督查管理、政务信息采集与发布、内部请示报告管理、档案管理、会议管理、领导活动管理、政策法规库、内部论坛等应用。其中涉及到了很多种表，如部门名称表、车辆管理表、收文登记表、发文登记表等。这些表里面到底有哪些字段，每个字段应该符合什么样的需要，这些需求很难确定，用户往往是等设计出来软件使用以后才能逐渐确定好需求。因此在使用 ORM 组件以前，开发人员不得不一次又一次地修改数据库的表结构，然后修改 SQL 语句，修改数据访问层的代码，再重新编译、调试，并且在修改代码的过程中还经常会出现错误，浪费了大量的人力物力和时间。

2.2 在办公自动化系统中应用 ORM 组件的考虑

由于目前所有关于对象—关系映射的研究都没能提供关于这种映射是否完备的证明，所以实现完全意义上的对象—关系映射是非常困难的，以至 Scott W. Ambler 在文献[2]提出了警告："构建一个持久层惊人的困难"。

Martin Fowler 在文献[3]中，把对象关系映射当作了一个设计模式加以叙述和论证，并对其他相关子模式也进行了详细的阐述。该书可以算是对近年来的 ORM 的研究成果和技术作了一个总结。在书中他提到："我认为模式是一种半生不熟品，为了用好它，必须在自己的项目中把剩下的那一半火候补上"。

因此，目前虽然已经存在很多的基于.NET 平台的 ORM 组件可以解决关系模型和数据模型之间"阻抗不匹配"带来的问题，比如 NHibernate，XPO，DataObjects 等组件。但是由于它们都是基于通用的目标设计的，试图解决几乎所有的问题，这使得它们结构太过于庞大、复杂而不灵活，也使它们短期内难以成熟到可以让开发人员放心地在项目中使用。如 NHibernate 虽然功能很强大，支持各种关系模式，而且还有比较灵活的 HQL 查询语言。但是它仍然脱离不了繁琐的关系，仍然对数据库不能完全的支持。而且还要重新学习 HQL 查询语言，如果不是经验丰富的专业人员将很难使用[4]。

ORM 组件实现的功能越强，相应带来的复杂度就越大，因此不能为了一些对实际项目不重要的功能或者很少用到的东西而把 ORM 组件设计得过于复杂，必须根据实际的需要在功能与复杂度之间做一个权衡。办公自动化系统中的大多数表报表都是独立的一张表，表与表之间没有复杂的关系，也很少用到复杂操作。大多数只是基本的增、删、更新和简单的条件查询。基于这种情况，笔者设计并实现了一个轻量级的 ORM 组件——MY ORM LAYER，该组件能满足办公自动化系统中大多数应用的需要。

2.3 MY ORM LAYER 的功能

该组件能根据映射文件自动生成增、删、更新等这些对象的基本方法来对数据库进行操作。并且能根据用户对搜索条件的选择构造出形如 Select * from table where id=A OR name=B AND partment=C 这样的查询语句直接送入数据库执行来实现简单的条件查询。因此该组件能使开发人员避免为了一些简单而又常用的基本操作而写 INSERT，UPDATE，DELETE，SELECT 语句。解决了编写这些 SQL 语句造成的开发速度慢、容易出错等问题。对于复杂的操作，如视图的查询、存储过程等，该组件提供 SQL 接口，由用户自行编写 SQL 语句通过该接口执行。

2.4 架构的设计

MY ORM LAYER 组件体系结构如图 1 所示。

对于数据库中的每个表，MY ORM LAYER 在 Mapping Schema 模块中使用以表的名字命名的 XML 文件来描述表结构以及表结构和实体类结构的对应关系，再由 Object Mappingmo 模块生成对应的包含 Addnew()，update()，delete()，find()这样的对象方法的实体类；最后通过 Osql Engine 将这些对象方法转换成对应的 SQL 语句送到数据库中执行。这样就把业务系统中对实体类的操作映射到了数据库中，完成了真正的数据库操作。由于实现的功能比较简单，MY ORM LAYER 不用支持 Relations，这使得它在能满足办公自动化系统的基本需求的基础上，比其他的 ORM 组件要容易上手，容易理解。

MY ORM LAYER 内封装了 Simple Query 模块,Web 端调用这个模块获得对应表的字段信息并生成一个带简单条件逻辑(支持并、或以及基本的逻辑门)的查询页面。

对于复杂的查询,组件提供了 SQL 的接口——Sql Interface,以便用户直接使用 SQL 语句来进行一些复杂的数据操作。

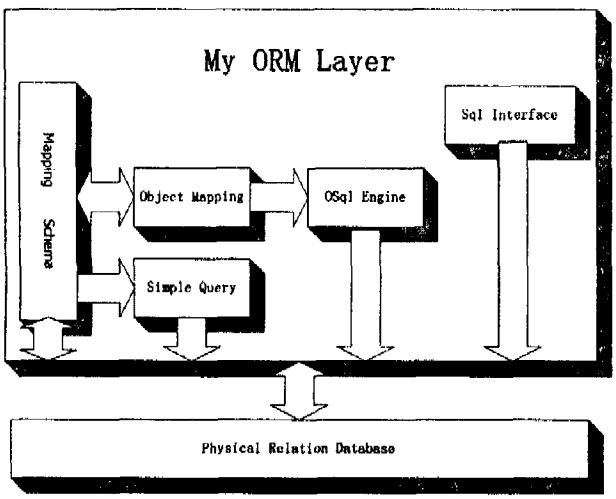

图 1　体系结构图

2.5　组件的使用

以一个部门表为例,该部门表的名称是 DepartMent,表里面有 3 列——自增的主键'ID'、部门名称'DepartMentName'、部门人数'DepartPepleNum',类型分别为 int,nvarchar(64),int。

相对应的 XML 文件如下所示,其中 column 表示的是数据库表中的列名,property name 则是列在类中对应的属性名称。

```
<? xml version="1.0" encoding="utf-8" ? >
  <MyORMLayer-mapping xmlns="urn:MyORMLayer-mapping-2.0">
  <class name="DepartMent " tabme="DepartMent">
  <id name="ID" column="ID" type="int">
  <generator class="assigned"/></id>
  <property name="DepartMentName" type="Nvarchar(64)" column="DepartMentName"/>
  <property name="DepartPepleNum" type="Int(8)" column="DepartPepleNum"/>
  </class>
</MyORMLayer-mapping>
```

组件根据 XML 文件产生的实体类如下:该实体类包含 3 个属性——ID,DepartMentName,DepartPepleNum,类型分别为 int,string,int,包含 4 个方法——Addnew()、update()、delete()和 find(),通过调用 Osql Engine 将这些对象方法转换成对应的 SQL 语句送到数据库中执行。

```
using System;
using Osql_Engine:
namespace DepartMent
{
public class DepartMent
 {
   public DepartMent()
   {
   }
   private System.Int32 ID;
   public System.Int32 ID
   {
     get {return ID;}
     set {_ID=value;}
   }
   private System.String DepartMentName;
   public System.String DepartMentName
   {
     get {return DepartMentName;}
     set {_DepartMentName=value;}
   }
   private System.Int32 DepartPepleNum;
   public System.Int32 DepartPepleNum
   {
     get {return DepartPepleNum;}
     set {_DepartPepleNum=value;}
   }
   public void Addnew()
   {
     Osql_Engine.TraslateToSql("Add","String","String",DepartMentName,DepartPepleNum);
   }
     public void Update()
   {
      Osql_Engine.TraslateToSql("Update","Int","String","String",ID,DepartMentName,DepartPepleNum);
   }
   public void Delete()
   {
     Osql_Engine.TraslateToSql("Delete","Int",ID,);
   }
   public DepartMent Find(int id)
   {
     Osql_Engine.TraslateToSql("Find","Int",id);
   }
 }
}
```

在.NET(C#)程序中,使用该组件操作数据库的方法如下:

```
// 向数据库添加数据
DepartMent dep=new DepartMent();
dep.DepartMentName="办公室";
dep.DepartPepleNum=21;
dep.AddNew();
//修改数据,修改前应该先得到序列号,也就是 ID 列
```

```
DepartMent dep=DepartMent.find(ID);
dep.Name="保密室";
dep.DepartPepleNum=22;
dep.Update();
// 删除数据
dep.delete(ID);
```

这样的代码里面没有任何一句 SQL 语句,达到了对数据层透明,提高软件开发效率的目标。Web 端调用 Simple Query 模块实现简单查询方式如图 2 所示。

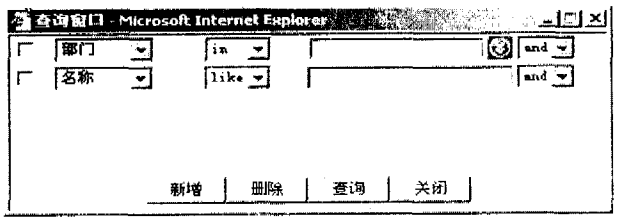

图 2 简单查询方式

根据用户对逻辑条件的选择,构造出 sql 语句直接送入数据库执行来实现简单的条件查询。通过采用 MY ORM Layer,使得当数据库中的表结构发生变化时,开发人员只需要修改对应的 XML 映射文件而不必改动代码,提高了系统的应用性能及可维护性;并且不必要为了一些简单的操作编写代码,节约了开发成本和时间。

3 ORM 组件的未来

随着 ORM 组件的数据缓存、延迟加载、SQL 语句优化及批量执行等技术的成熟,它的性能和直接访问数据库方式的性能差距将越来越小。并且随着计算机硬件性能的快速提高,性能将不再是 ORM 组件的发展瓶颈。未来的 ORM 组件将向着和 VS.NET 等开发环境集成,通过图形化界面定义类结构、映射关系、自动生成相关的实体类和 XML 映射文件等方向发展,并且类似 XPO 一样支持根据 class 直接生成数据库 schema 功能的 ORM 组件将成为主流。这类 Class→DB 的 ORM 组件将能和 UML 等建模工具集成,根据建模工具设计的类结构自动生成数据库表结构。这就不仅将程序开发者从繁琐的数据库设计中解脱出来,而且还从业务逻辑代码的编写中解脱出来,使他们能真正以面向对象的层次设计软件而无需关心底层的实现细节[5]。

参考文献:

[1] 飞 鹰. 什么是 O/R Mapping? [EB/OL]. http://blog.aspcool.com/tim/archive/2004/09/17/1109.aspx., 2004-09-17.

[2] Ambler S W. Robust Persistence Layer For Relational Databases [EB/OL]. http://www.ambysoft.com/persistenceLayer.pdf,1999-10-09.

[3] Fowler M. 企业应用架构模式[M]. 王怀民,周 斌,译. 北京:机械工业出版社,2004.

[4] Open Source nhibernate development group. What is NHibernate? [EB/OL]. http://nhibernate.sourceforge.net/, 2003-02-11.

[5] Progame. 功夫、Persistore 及其它[EB/OL]. http://www.cnblogs.com/progame/archive/2004/12/26/82133.html, 2004-12-26.

(上接第 7 页)

5 结束语

文中采用的过滤模型由于建立在 WCCP 框架中的缓存服务器上,所以充分利用了 WCCP 的优点——对于用户的重复请求利用缓冲机制予以处理,同时信息过滤可以在用户完全不知觉的情况下进行。

该模型在过滤信息上具有以下 3 个特点。首先是采用了两层过滤方式,传统的 URL 过滤方式需要专人维护非法 URL 表,而且对非法 URL 的覆盖率也不高,传统的内容过滤方式对每次用户请求后从 Internet 上得到的页面都要进行一遍内容过滤,拖延了请求响应时间,而文中提出的过滤模型避免了 URL 过滤与内容过滤的缺点,同时也实现了 URL 过滤的高效率和内容过滤的全面性,具有较好的性能;其次在 URL 过滤上,传统的 URL 过滤方法只是将用户请求中的 URL 与数据库中的非法 URL 表中的记录进行简单的比较,效率偏低,而该模型结合了缓存机制及合法 URL 和非法 URL 双表过滤机制,提高了 URL 匹配的速度,进一步优化了 URL 过滤性能;最后对于内容过滤该模型也进行了改进,针对传统内容过滤大多进行的是文本过滤,对信息的过滤不够完全,该模型采用了网络爬虫分析网页确定该网页的向量特征的方法,对网页合法性的确定更加合理。

参考文献:

[1] Sripada S, Reiter E, Hunter J, et al. A Two-stage Model for Content Determination[A]. In Proceedings of the 8th ACL-EWNLG'2001[C]. Toulouse, France: [s.n.], 2001. 3-10.

[2] Sublime Solutions Pty Ltd. Squid Cache Transparency and WCCP Version 1.1 [EB/OL]. http://www.sublime.com.au/squid-wccp. 2001.

[3] Wessels, Duane. Web 缓存(英文影印版)[M]. 北京: 清华大学出版社, 2002.

[4] Hammami M, Chahir Y, Chen Liming. WebGuard: Web Based Adult Content Detection and Filtering System [A]. In Proceedings of the IEEE/WIC International conference on web Intelligence (WI'03)[C]. Halifax, Canada: [s.n.], 2003. 574-578.

[5] Oard D W, Marchionini G. A Conceptual Framework for Text Filtering[Z]. Technical article CS-TR-3643. Maryland: University of Maryland, 1996.