.NET 下的数据访问技术

朱永光

自我介绍

- 自由人，开发技术咨询，解决方案构架
- 编程实践13年，从业6年，4年构架经验
- 擅长微软相关技术和产品，对其他技术也有涉及，对语言和操作系统比较感兴趣
- 目前主要关注软件构架和开发框架

议题

- 数据访问技术
- ADO.NET
- ORM for .NET
- 对象数据库
- 数据库访问代码生成工具
- 其他技术和工具

数据访问技术发展

- 从数据库技术诞生到现在，已经有了数十年的历史；相应的，从最早的读写磁盘文件开始，数据访问技术也不断的推陈出新，变得越来越丰富。
- 最早，不同的数据库有不同的访问接口
- 接着，出现了最初的 ODBC及类似的访问技术
- 后来，上述技术在微软的推动下不断发展，后来又有了诸如 OLE DB（Object Link and Embedding Database）以及 ADO（ActiveX Data Object）等数据访问技术

主流数据访问技术

- **JDBC（Java Database Connectivity：Java 数据库连接）**
  - Type 1, Type 2, Type 3, Type 4
- 在主流的数据访问技术中，微软相关的数据访问技术自成一套体系。
  - ODBC（Open Database Connectivity）
  - DAO（Data Access Objects）
  - RDO（Remote Data Object）
  - OLE DB
  - ADO
  - ADO.NET

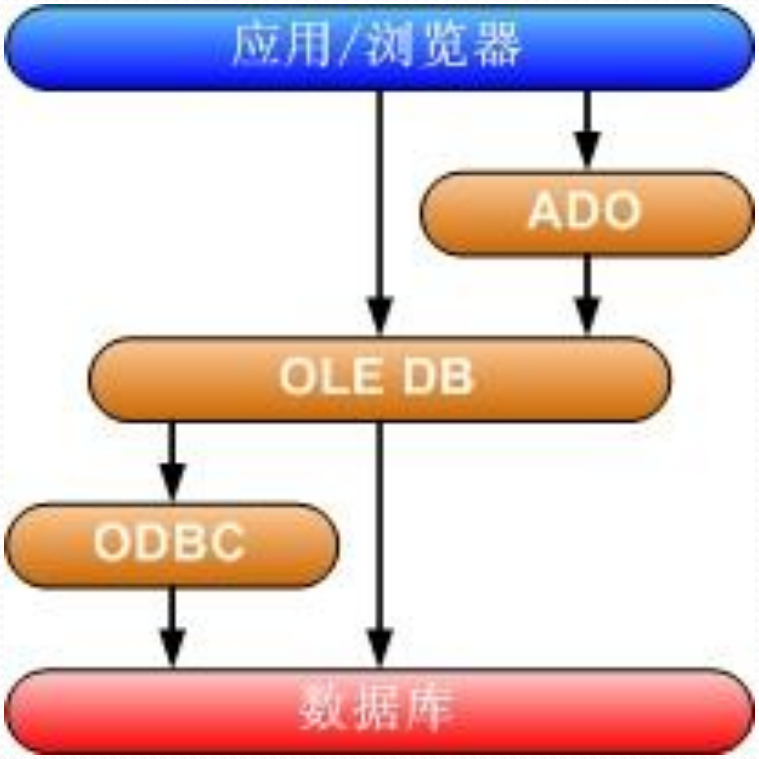
数据访问模式

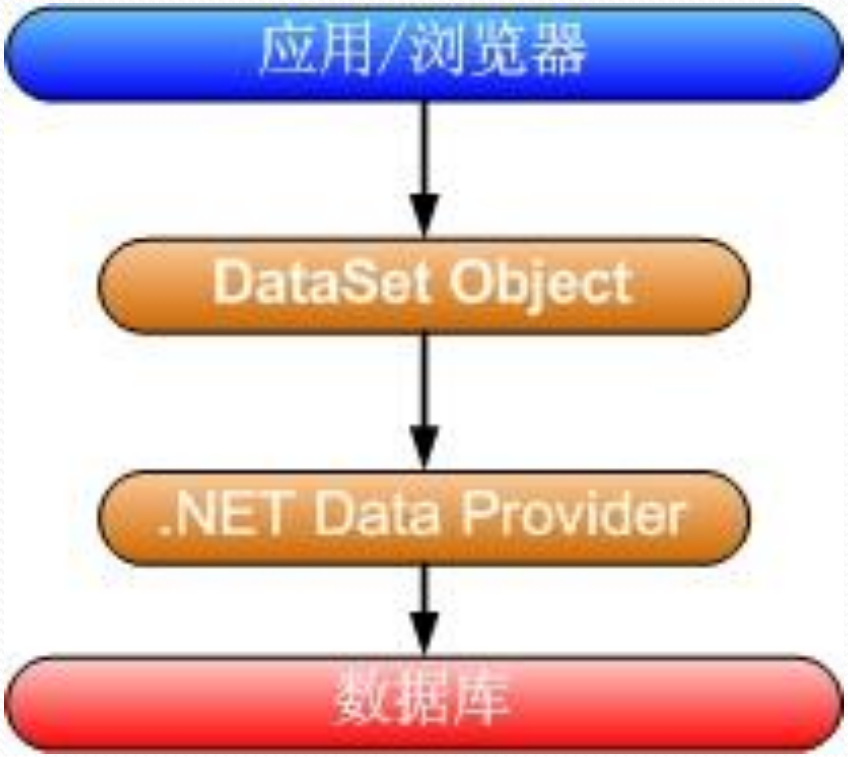
- 在线访问
  - 会占用一个数据库连接，读取数据，每个数据库操作都会通过这个连接不断地与后台的数据源进行交互
- Data Access Object
  - DAO 模式是标准 J2EE 设计模式之一，开发人员常常用这种模式将底层数据访问操作与高层业务逻辑分离开
- Data Transfer Object
  - Data Transfer Object 是经典 EJB 设计模式之一，DTO 本身是这样一组对象或是数据的容器，它需要跨不同的进程或是网络的边界来传输数据

数据访问模式

- 离线数据模式
  - 以数据为中心，离线，与 XML 集成，独立于数据源
  - WebSphere 平台中的实现 - SDO
  - JDBC v3.0 中的 CachedRowSet
  - 微软 .NET 框架下的 ADO.NET 技术
- 对象/关系映射(O/R Mapping: Object/Relation Mapping)
  - 提供一种工具或是平台，能够帮助将应用程序中的数据转换成关系型数据库中的记录；或是将关系数据库中的记录转换成应用程序中代码便于操作的对象。

ADO vs ADO.NET





ADO Code

```
<!--#include file= “adovbs.inc” -->
<%
Dim connStr, rs
connStr = “Provider=SQLOLEDB.1;Persist Security
  Info=False;User ID=sa;Initial Catalog=pubs;Data
  Source=localhost”
SET rs= Server.CreateObject( “ADODB.Recordset “)
rs.Open “Authors”, connStr, adOpenForwardOnly,
  adLockOptimistic, adCmdTable
WHILE NOT rs.EOF
  response.write rs( “au_fname” ) & “,” & rs( “au_lname” ) &
   “<br>”
  rs.moveNext
END
SET rs=nothing %>
```

ADO.NET Code

```
Dim sql AS String = “SELECT * FROM Authors”
Dim conn AS New SqlConnection(“server=localhost;
   uid=sa; password=; database=pubs”)
Dim comm AS New SqlCommand(sql, conn)
Dim DataAdapter AS New SqlDataAdapter(comm)
Dim ds AS New DataSet()
conn.Open()
DataAdapter.Fill(ds, “Authors_table”)
conn.Close()
```

ADO.NET简介

- 全新设计的数据访问框架
- 满足了 三个重要需求：断开的数据访问模型；与 XML 的紧密集成；与 .NET 框架的无缝集成。

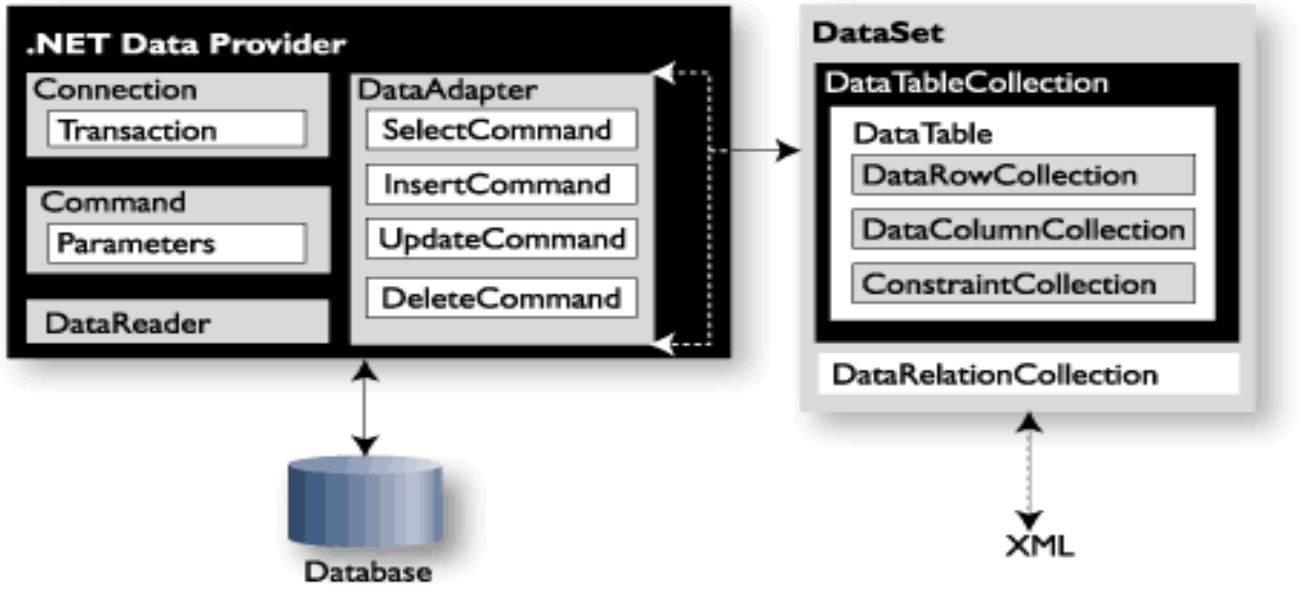


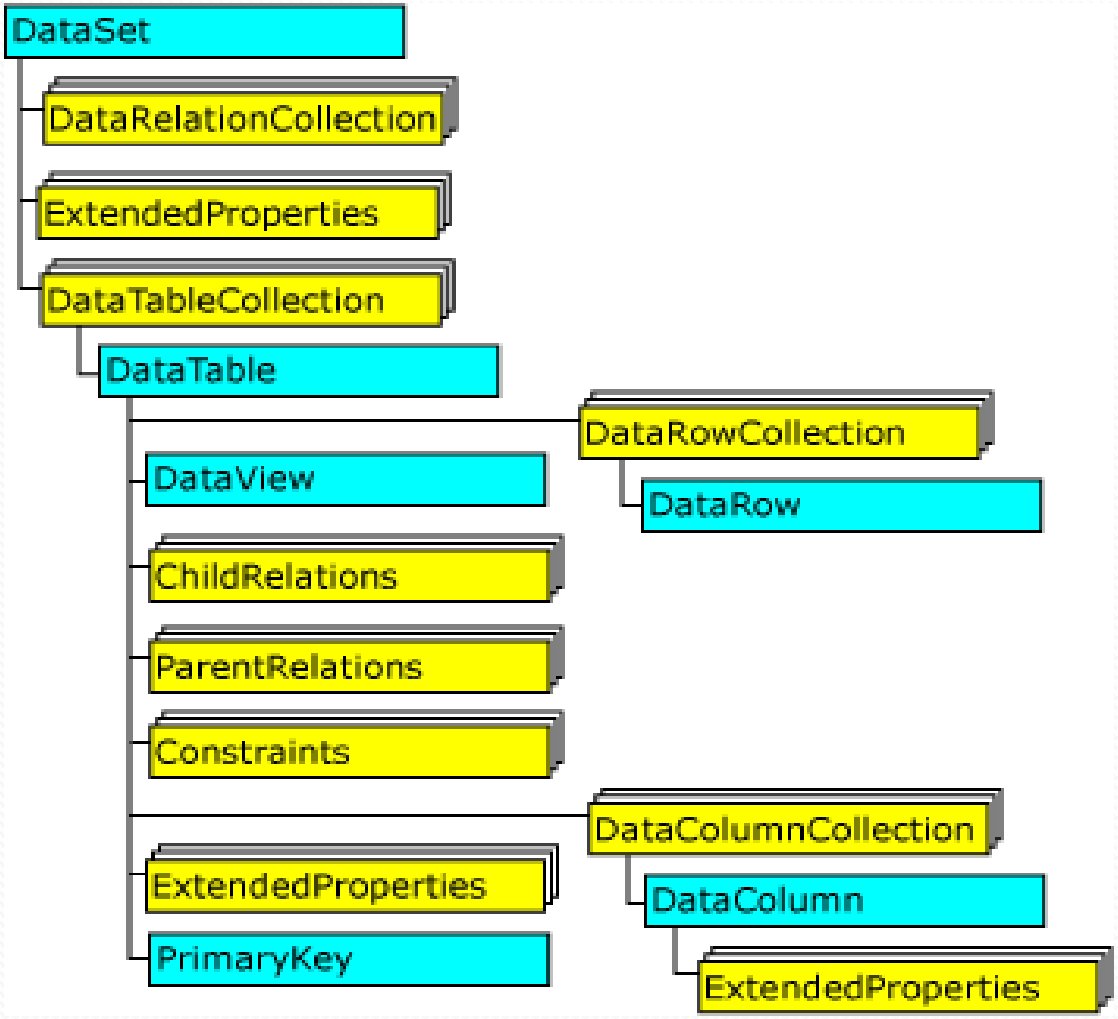
FIGURE 7: ADO.NET architecture

ADO.NET Data Provider

- Connection 对象，用于连接数据源;
- Command 对象，对数据源执行命令;
- DataReader 对象，在只读和只进的连接模式下从数据源读取数据;
- DataAdapter 对象，从数据源读取数据并使用所读取的数据填充数据集对象
- SQL Server .NET Framework 数据提供程序
- OLE DB .NET Framework 数据提供程序
- ODBC .NET Framework 数据提供程序
- Oracle .NET Framework 数据提供程序

ADO.NET DataSet

- DataSet 对象是支持 ADO.NET 的断开式、分布式数据方案的核心对象。DataSet 是数据的内存驻留表示形式，无论数据源是什么，它都会提供一致的关系编程模型



ADO.NET 2.0

- 异步处理
- 针对Microsoft SQL Server 2005的新特性
- 批处理
- DataSet 和DataTable 增强
  - DataTableReader
  - 新的索引引擎
  - 二进制序列化
  - DataTable 作为独立对象
  - 从 DataView 创建 DataTable
  - 行状态控制:新的 SetAdded 和 SetModified 方法

ADO.NET 2.0(与提供程序无关)

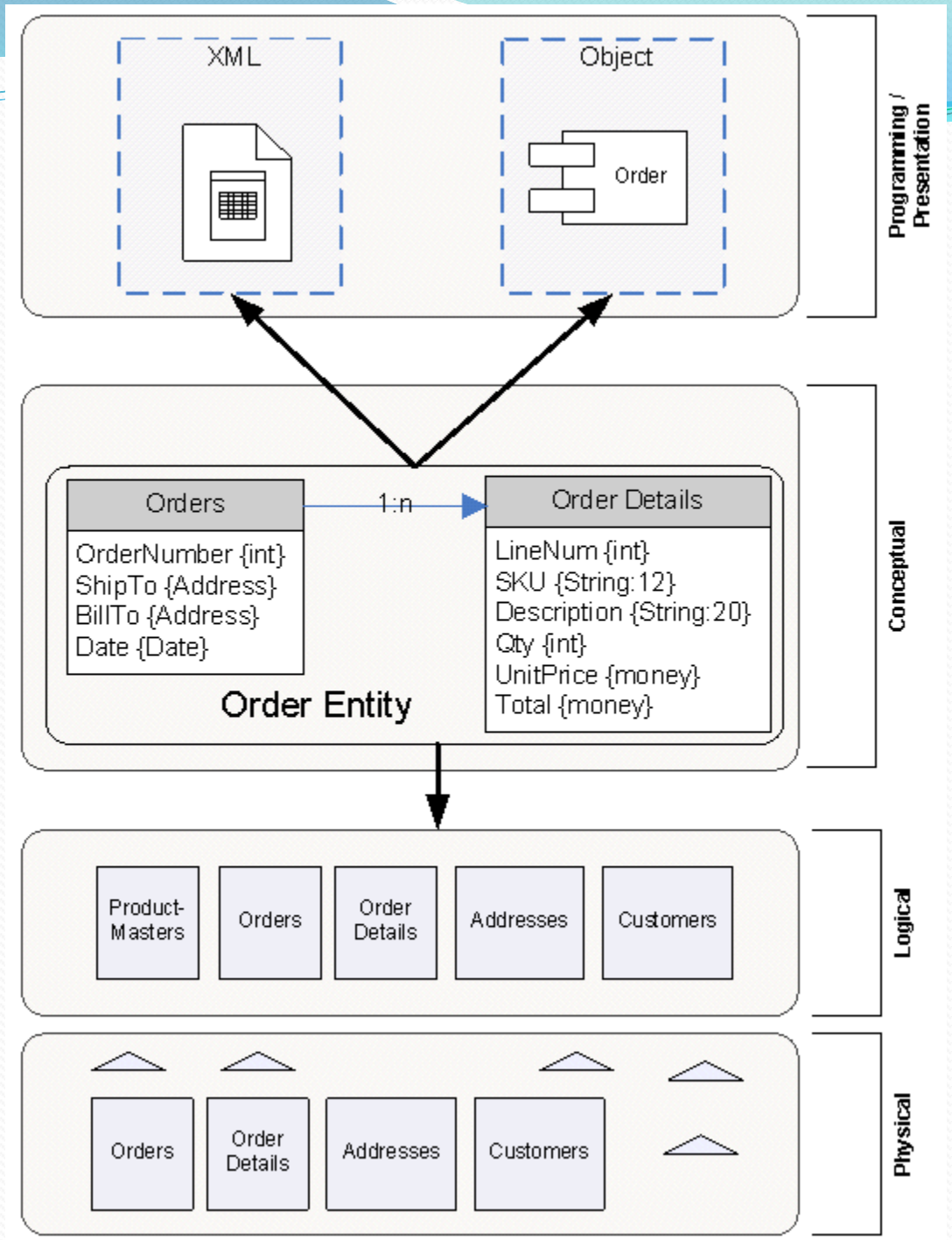
- **System.Data.Common 命名空间**
- 获取 DbProviderFactory
- 创建命令和检索数据
- 使用 DbDataAdapter 检索数据

Visual Studio 2005 for ADO.NET 2.0

- “数据源”窗口提供拖放数据绑定
- 用于轻松连接数据的数据源配置向导
- 绑定到对象
- 数据集设计器
- TableAdapter
- TableAdapter 查询配置向导
- 在应用程序设置文件中保存连接字符串
- 本地数据
- 强类型数据集和 TableAdapter 的分部类支持
- Windows 窗体应用程序
- 数据智能标记
- 在托管代码中创建 SQL Server 2005 数据库对象
- 可视化数据库工具

下一代ADO.NET

- 下一代数据访问平台所处理的一个关键问题就是众所周知的应用程序阻抗失谐问题
- 通过将抽象级别从逻辑（关系）级别提高到概念（实体）级别来消除应用程序和数据服务（例如，作为 SQL Server 产品一部分提供的报告、分析和复制服务）两方面的阻抗失谐


XML
Object
Order
Programming / Presentation
Orders
1:n
Order Details
OrderNumber {int}
ShipTo {Address}
BillTo {Address}
Date {Date}
LineNum {int}
SKU {String:12}
Description {String:20}
Qty {int}
UnitPrice {money}
Total {money}
Order Entity
Conceptual
Product-Masters
Orders
Order Details
Addresses
Customers
Logical
Orders
Order Details
Addresses
Customers
Physical

ADO.NET实体框架

- 实体数据模型（Entity Data Model):
  - 是一个“实体-关系”数据模型
  - EDM 中的核心概念是实体和关系
  - 实体是“实体类型”的实例（例如 Customer、Employee），即含有一个键值的充分结构化的记录。
  - 关系是“关系类型”的实例，是两个或多个实体类型之间的关联（例如 Employee WorksFor Department）
- ADO.NET实体框架(Entity Framework):该框架不是一个全新的,独立的基础结构。他只是在我们所了解的传统Ado.net上提供一种新的选择

ADO.NET实体框架 分层

存储提供程序(Storage Provider)

该层由数据访问提供程序(data-access providers)组成,进行数据源的访问和通讯,例如:微软SQL Server或者Oracle(例如SqlClient,OracleClient等等).

ADO.NET实体框架 分层

映射层(Mapping Layer)

该层处于EDM之上,对ADO.NET中API的概念进行一个映射.这个层中有如下顶层类:MappingProvider,MapConnection和MapCommand(他们的名字可能在今后的工作过程中被更改).EDM模型为你的应用程序提供一个概念上的视图,该视图表达特定领域中的数据.映射提供程序(mapping provider)会得到EDM的架构和映射信息,因此他可以在其内部使用映射架构在概念和逻辑架构之间进行转换.通过使用EDM模型和映射提供程序(mapping provider),你的应用程序不再使用或者看到数据库相关的结构.整个应用程序将在更高级别的EDM模型的基础上进行操作.

这同样意味着你不能再使用原生的数据库查询语言(SQL-92),取而代之将是实体SQL(Entity SQL).Entity SQL是专门为实体数据模型(Entity Data Model)而设计的,他对EDM有完整的表达性(fully leverage the expressivity).Entity SQL的查询可以在设计时静态的表述,也可以在运行时构建.

ADO.NET实体框架 分层

对象层(Object Service)

该层的目标是消除数据与应用程序代码间的失谐(阻抗失谐).通常的Ado.net将数据库中的数据以行和列的形式暴露出来,现在同样的数据将以对象的形式暴露出来(至少现在我们有了这个选择).该层同样包含更多的高级服务,这些服务通常支持ORM框架中的诸如标识(identity),变动追踪(Change tracking),乐观并发检测(check for optimistic concurrency)和更新操作(暂不明确).

ADO.NET实体框架 分层

LINQ to Entities

该层将LINQ项目和实体框架集成起来,这样能在更高级别的面向对象编程语言(例如C#)上进行自然表达式的查询.LINQ to Entities层依赖于对象服务层(object services)和映射层(mapping layer).LINQ查询被转换为规范的查询树(query trees),转换结果和解析Entity SQL语句一样,然后被映射层处理.

什么是Object Relational Mapping（ORM）

- ORM用于解决“阻抗匹配”，“阻抗失谐”
  - 数据库的焦点在于表，行，索引和基于键的关系
  - 对象的焦点在于对象图，继承，多态和属性，及基于对象的关系
- 但是数据库和对象之间不能很好的协调
- 所以ORM就是用于解决这个Gap的
  - 提供一种方法自动映射对象到数据库，反之也然
  - 可以从简单处理单表到单对象，也可以支持多对象到多表的映射，并提供对象缓存等特性

ORM应该具有的一些特性

- 所有的关系类型（1-1，1-n，n-n）
- 事务
- 映射单对象到多表，反之也然
- 对象继承
- 对象缓存
- 优化查询
- 延后加载
- 支持多种RDBMSs
- 具有管理的图形界面
- 在内存中支持过滤（避免对数据库的往返操作）
- 对象查询语言
- 支持混合的键
- 支持多种类型主键（自增，Guid，等等）

ORM的一些实现方式和例子

- 代码生成
  - LLBLGen Pro
  - Wilson ORMapper
  - CodeSmith
  - MyGeneration
  - Codus(Demo)
- Attributes
  - Gentle.NET
  - Castle ActiveRecord(Demo)
- XML
  - NHibernate

对象数据库ODBMS

- 是一种以对象形式表示信息的数据库，直接把对象保存在数据库中，并以特定的对象查询语言来操作数据
- 关系数据库在管理复杂数据时显得笨重
- 解决“阻抗匹配”
- DB4O(Demo)
  - 完全原生于Java和.NET 100% 面向对象,
  - 抛开对象-关系映射
  - 为嵌入式应用优化
  - 开源,可以基于GPL协议免费使用.
- ZODB
  - The Z Object Database, or the ZODB for short, is a powerful and easy-to-use object persistence system for Python

Base4.NET

- Base4 is a powerful open source platform and API for extending and integrating existing legacy and 3rd party databases
- is much more than an OR mapper
- Top 5 reasons why Base4 is different:
  - **(1) Simplicity**
  - **(2) Data Model Re-use**
  - **(3) Integration**
  - **(4) Files**
  - **(5) Meta-data**
- (Demo)

Q & A

Thanks

- Blogs: http://redmoon.cnblogs.com/
- MSN: heavenwing@msn.com
- Email: redmoon17@gmail

参考文献

- 选择数据访问模式，合理规划数据访问层
- 数据访问技术的演变
- .NET 数据访问架构指南
- 下一代数据访问：使概念级别成为现实
- ADO.NET 技术预览：实体数据模型
- ADO.NET Entity Framework Layering