TURING 图灵程序设计丛书 .NET系列

Apress®

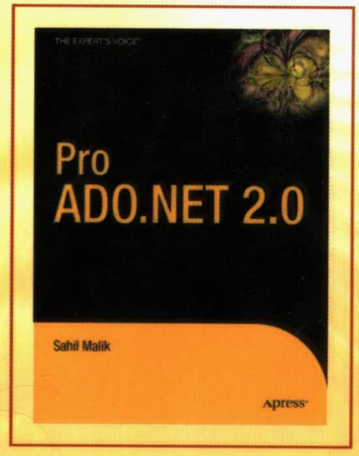
Pro ADO.NET 2.0

ADO.NET 2.0 高级程序设计

[美] Sahil Malik 著

汤涛 邰晓翠 译

- 微软 MVP 力作
- 涵盖 ADO.NET 2.0 的方方面面
- 同时提供 C# 和 VB.NET 代码



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 .NET系列

ADO.NET 2.0高级程序设计

Pro ADO.NET 2.0

[美] Sahil Malik 著

汤 涛 邰晓翠 译

人民邮电出版社

北 京

图书在版编目（CIP）数据

ADO.NET 2.0 高级程序设计 /（美）马利克，（Malik,S.）著；汤涛，邰晓翠译.
—北京：人民邮电出版社，2007.6
（图灵程序设计丛书）

ISBN 978-7-115-15868-0

Ⅰ. A… Ⅱ. ①马…②汤…③邰… Ⅲ. 软件工具—程序设计 Ⅳ. TP311.56

中国版本图书馆 CIP 数据核字（2007）第 024184 号

内 容 提 要

本书是介绍 ADO.NET 2.0 的权威参考书，用 VB.NET 和 C# 两种语言来描述，详细讲解了与数据库的连接、获取数据以及与事务工作的实际操作，而不是简单重复 MSDN 的文档。本书提供了深入的理解、全面的观点，还解释了如何用 ADO.NET 的技术来解决实际问题和搭建应用程序的架构。

本书适用于中、高级的 .NET 应用开发人员。

图灵程序设计丛书

ADO.NET 2.0 高级程序设计

◆ 著 [美] Sahil Malik
译 汤 涛 邰晓翠
责任编辑 陈兴璐

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：800 × 1000 1/16
印张：29.25
字数：685 千字 2007 年 6 月第 1 版
印数：1–4 000 册 2007 年 6 月北京第 1 次印刷

著作权合同登记号 图字：01-2006-3206 号

ISBN 978-7-115-15868-0/TP

定价：59.00 元

读者服务热线：(010)88593802 印装质量热线：(010)67129223

版 权 声 明

谨将本书献给我的父母。

献给妈妈，

自我孩提时起，她就非常严格地要求我，并且在我生命中源源不断地给予了爱。

献给爸爸，

他把我从妈妈手中拯救出来☺，而且是我灵感和力量的无尽源泉。

我非常爱你们！

前　言

……请核对任务……你现在正在接近微软技术银河.NET 恒星系中的 ADO.NET 星球。请确保没有架构错误让外星人把你当晚餐给吃了……[1]

学习任何新鲜事物就像接近一个新的星球。由于你是从遥远的地方飞近星球，你首先辨认出它在恒星系中的位置，接着是表面的主要特征，最后你登陆该星球开始挖洞、盖房子，建设家园。然后，在你尚未了解这个星球以前，你就娶了一位外星夫人，生了两个小孩，负担一处房产按揭，一辆汽车的按揭贷款，最后开始担心孩子的大学教育基金。

这是真的！生活就像计算机游戏，它会变得越来越艰难。

学习 ADO.NET 也是如此，从简单入手，到复杂结束。

这本书从三个比较短（合起来不到 50 页）而且比较简单的章节开始：

- 第 1 章标明 ADO.NET 在.NET 恒星系中的位置，及其主要的组成。
- 第 2 章识别 ADO.NET 地形的主要地理特征。它也是在未来章节中登陆后并开始深挖时所准备的导向图。因为这一章主要是导向图，所以会在本章提示你在深挖时可以参考的各种图表、类名和命名空间。
- 第3章是当你登陆星球并且开始到处走走时，创建四个稍复杂一些的数据驱动应用程序。

只要你登陆这个星球，装备了它的地图，考察周围环境后，就要开始深入挖掘了，就像人类对新事物的本能探索一样（希望不要使用爆破）。

有一个问题，当你手里拿着一个锤子，你会怎么做？你会用力敲打钉子头，对吧？

现在如果有人告诉你，这儿有一个锤子，包括锤头和锤柄。锤柄很长，它能帮助产生较大力矩，因为力矩与力矩臂的长度是成正比的。锤头越重，力矩能转换的动能就越大。现在因为根据如下等式，动能既不能消失也不能增加：

```
M1V1 = M2V2
```

而且钉子的质量很小，动能传递给钉子，从而让钉子产生非常高的速度，这样就钉到木头里去了。

哦，天哪！在听了对一个这么简单的问题这么复杂的描述后，我感觉就像在锤打我自己一样。为什么我们不能只说“锤子敲钉子头，因此钉子就钉到木头中去了”呢？这样更简单，不是吗？

那么，学习 ADO.NET 也可以像这样简单。这里有一些基本的操作是这种数据访问架构可以实现的：连接到数据库、获取数据、保存非连接数据、处理非连接数据，最后再将数据存回到数

1. 此处作者虚构了乘飞船登陆外星时的场景。——编者注

据库。在写作本书时，我就尽量专注于所需要完成的任务，并且尽量用这样的术语简化 ADO.NET 架构。

接下来是在 C#和 VB.NET、不同的数据库如 SQL Server 和 Oracle 之间做选择。这种选择其实意义不大，因为任何一方并不比另一方更有优势。因此我干脆不做选择，所有的例子都用 C#和 VB.NET 做演示。这些例子都可以很好地运行在 SQL Server 2005 上，同时也会注明访问 Oracle 的代码片断。一个很好的例子就是 MARS。它既可以访问 SQL Server，又可以访问 Oracle，当然我会有所侧重。为了不至于写成个大杂烩，我试图更专注于 SQL Server，但是也没有忽略 Oracle。

因此，从第 4 章到第 11 章的内容完全是与数据库无关的，采用了简单的任务导向的方法。也就是说我们不是简单地罗列 DbCommand 的方法列表，而是通过这样的方式讲述："你可能需要一个返回单个结果的查询，或是返回一行记录的，或是填充一个 DataSet 的。"

我可以就此结束本书。但是一本介绍 ADO.NET 的书，如果不涉及 SQL Server 2005 诸如 SQLCLR（SQL Server 内置的 CLR）和 XML 这样的特定功能，它就不够完整。因此，第 12 章和第 13 章留给 SQL Server 2005，专门来讨论这些问题的。

最后，架构（特别是数据访问架构）是一种充满挑战的技术，有大量需要进一步探究的灰色地带。本书最后一章讨论了通常围绕数据访问和应用程序架构的主要争议。

希望本书能为你提供足够的知识，从而能让你自信地应对各种架构性决策。

希望你能喜欢本书。

……再次核对任务……ADO.NET 星球已经看见了，掌好舵，在驾驶席上坐好，系好安全带，包好背包，然后翻到第 1 章。有趣的历程即将开始……

致谢

没有人是一个孤岛，也没有谁的思想会是一个孤岛。我只是一个传播者，把从我工作的领域及其他领域内很有造诣的专家那里听到的、读到的、学到的写出来而已。

我首先要感谢我的伙伴，Bill Vaughn、Bill Ryan、Miha、Herfried Wagner、Jon Skeet、Carl Franklin 以及其他优秀的精英，感谢他们为传播知识所付出的时间和努力。他们坚信星星之火可以燎原。正是由于与他们连续不断地讨论，难以计数地社区交互，我才能集成此书。这些想法都不是我的，我没有创造 ADO.NET——我只是从这些杰出的人那里学会了它而已。

接着，我要感谢两位最关键的、最亲密的审稿人。Frans Bouma 和 Erick Sgarbi 在审校工作中非常认真，令我感到非常惊讶。他们通篇仔细审查，字斟句酌地充实并完善书中内容。我要感谢他俩，不仅仅是因为他们是本书的审稿人，更因为他们是一直为我鼓劲的好朋友。

接着，我要感谢微软的一些员工，他们耐心地解答了我无数的问题。他们根本没有义务来帮助一个几千英里之外的陌生人，但是他们会在几分钟内回复我凌晨三点钟发送的电子邮件。他们是真正热爱自己事业的人，并开发出了如此奇妙的编程平台。要感谢 Pablo Castro、Angel Saenz Badillos、Sushil Chordia、Andy Conrad、Mike Clark、Jim Johnson、Raphael Renous、Mark Ashton、Michael Rys、Chris Lee、Steve Lasker 以及我的 MVP 主管 Rafael Munoz（我想我一定还遗漏了一

些人)。

接着,我还要感谢我的老板,他鼓励并指导我,就像一个父亲引导儿子一般。我要感谢 Michael Arluk,他给我提供了坚实的后盾,并且鼓励我完成本书的写作,这真是一项艰苦而又费时的工作。我已经告诉他并且还要再次声明,如果没有他这本书就不可能完成。

最后,我要感谢我的父母,因为他们对我要求严格,而且他们是最慈祥的父母之一。他们教我自律、奉献和专注,而且还教我学会了在艰苦时期要坚忍不拔。

Sahil Malik

目　录

第1章 ADO.NET概述

计算机可以看作是存储和处理信息的机器。虽说并不是每个应用程序都有专门的程序来管理其信息的存储，但也很难想像一个计算机程序不能处理任何一种数据。某些应用程序，像微软的Word和写字板，能维护它们自己的数据。其他大多数专用的应用程序，特别是那些需要海量数据的程序，需要有专门的程序或架构运行在一台独立的机器上，这种专门的程序或架构就叫作*数据库*。

有些应用程序使用基于服务器的数据库架构，比如Oracle，微软的SQL Server，MySQL，DB2等，而另外一些应用程序则使用基于文件的架构，比如微软的Access或Excel。

可使用各种计算机程序更为有效地管理信息：那些专门处理信息的应用程序，比如数据库，它们处理数据的方式就与介于用户和数据库之间的程序完全不同。大多数数据库把信息存储在表中，即把数据作为表中的行、列和值的集合。当今很多数据库还可以在这些表之间设定关系，这样就可以在具有关系的不同数据表之间保持数据的完整性。

但是，编程语言具有不同的数据表示方法。特别地，现在很多面向对象的编程语言通过对象的层次结构来表示数据。

实际上，一个程序可以一次与多个数据源工作，这就需要某种数据访问类库来实现这个功能，如图1-1所示。

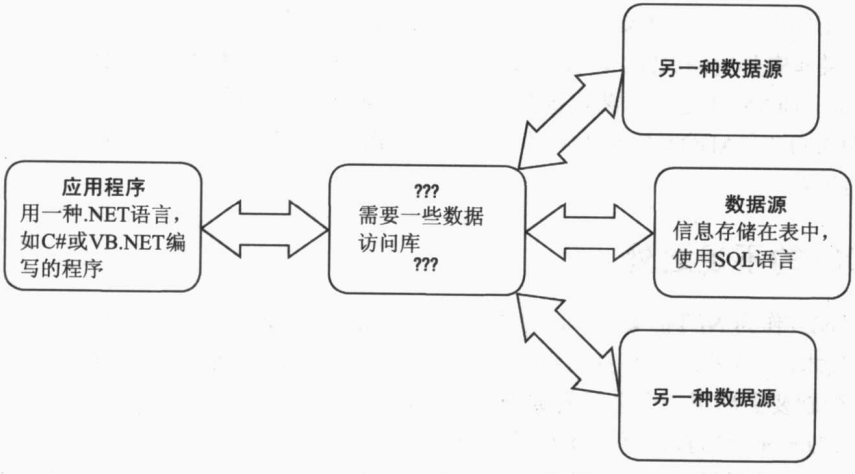


图1-1　一个典型的程序及其数据源

因此，大多数数据库处理信息的方式与很多编程语言处理信息的方式就会出现不匹配的情况。这恰好是ADO.NET的用武之地。

但什么是ADO.NET？

1.1　什么是ADO.NET

ADO.NET是微软.NET框架的一部分，它由一组工具和层组成，应用程序可以借此与基于文件或基于服务器的数据存储很轻松地进行通信和管理。在.NET框架中，ADO.NET类库位于System.Data命名空间下。这些类库包括连接到数据源、执行命令以及存储、操作和获取数据等功能，如图1-2所示。出于简化讨论的目的，在此仅列出一个数据源，但是一定要记住数据源可以是多个。

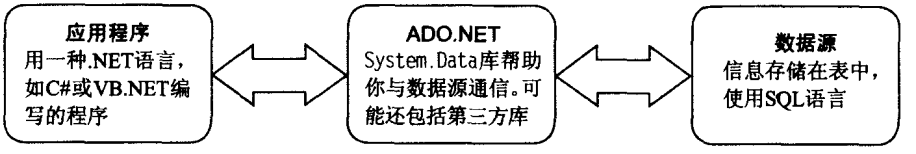


图1-2　什么是ADO.NET？它在图中什么位置比较合适

ADO.NET与以前的数据访问技术相比的不同之处，就是它可以让应用程序与数据库以完全非连接的数据缓存的方式来交互以实现离线操作数据。过去使用诸如远程数据对象（RDO）来实现这样的功能，但是那只不过是在已有的技术上做的一些修正。而ADO.NET则是基于这样的需求从头开始搭建的。

非连接的数据访问（disconnected data access）对于当今高要求的应用程序而言是很关键的，它并不是简单地直接把单个用户或系统中的实体连接到数据库。在高要求的场景下构建应用程序将在后面第4章进行深入讨论。

非连接的数据访问的一个要素就是用于表格数据的与数据库无关的容器。这个非连接的容器在ADO.NET类库中是用DataSet或DataTable对象来表示的。这些对象将在第6章中进一步讨论。

要知道在ADO.NET之前，微软曾有所谓经典ADO或就叫ADO这样一个早期的数据访问技术。虽然ADO.NET和ADO是完全不同的数据访问架构，但了解ADO哪些地方不足才能更好理解ADO.NET的长处。

1.2　ADO的不足之处

在ADO.NET作为.NET框架的一部分推出之前，ADO（ActiveX Data Object）是微软平台上的数据访问技术。一个很显而易见的问题是：为什么在大家都已经很熟悉的ADO能实现同样功能的情况下，微软要去启用一个全新的数据访问技术？推而广之，这个问题可以扩展为“DAO、RDO、ODBCDirect、OleDb以及ADO有什么不足之处？”

简言之，在过去的这些年里，数据访问的需求发生了变化，这样就需要对先前的数据访问技术做相应的调整。ADO主要通过非托管（unmanaged）代码访问数据，而在.NET中就要通过一个

标准的访问COM对象的机制，去编写非托管代码以访问ADO对象，这种访问COM对象的机制称为交互（interop）。与完全的托管代码相比，通过交互访问非托管代码时不仅会有性能的损耗，还会有悖于.NET的安全控制。并且，非托管代码会遇到原先在编程模型下的诸如DLL地狱（DLL Hell）之类的问题，针对基于交互的对象的垃圾收集实现得也不够理想。你可能已经了解到，在很大程度上，.NET下的垃圾收集能减轻每个程序清除其空闲内存的负担。这个功能在COM环境下不可用，需要依赖于引用计数器来实现内存管理。如果架构有问题，那么交互的COM组件就可能无法与新的垃圾收集模型一起工作，甚至可能导致内存泄漏。最后要注意的是在数据层的内存泄漏的问题，这通常是应用程序中最为关键的部分，它与性能和可靠性同样重要。

ADO的另一个大缺陷是，它在设计时没有考虑与XML一起工作，XML是后加的，而且那些使用过ADO的人都知道，虽然Recordset可以转换成XML，但所产生的XML是很难读懂的，而且在不同的对象之间并不能随心所欲地移植。相反，ADO.NET则是基于这些需求从头开始设计的。

按照同样的思路，在开发ADO时，Web服务和整个非连接的计算概念还处在早期。随着非连接计算和对集中式数据库的大量需求的激增，很显然需要一种新的数据访问架构——必须支持更好的并发、池化、XML支持以及非连接的架构。因此，ADO.NET应运而生。

1.3 ADO.NET 中的重要对象

与其他任何架构一样，ADO.NET也是由一些重要部分组成的。本节中，你将看到组成ADO.NET的各种对象。

你可能知道，.NET类可以通过命名空间组织到一起。所有ADO.NET相关功能的类都位于System.Data命名空间下。这并不是说其他软件开发人员不能编写不属于这个命名空间但又实现了数据访问功能的类库，但是到.NET框架发布时，属于微软.NET框架一部分的所有ADO.NET相关功能都位于System.Data命名空间下。

同样，就像其他.NET组件一样，ADO.NET并不是孤立的，它可以与.NET框架中各种其他组成部分——诸如System.Web.UI.WebControls.Adapters.TableAdapter类或System.Transactions命名空间等实现交互。

ADO.NET架构可以分成两个基本类别：连接的和非连接的。在ADO.NET中的各种类都可以分成连接的和非连接的。唯一的例外是DataAdapter对象，它充当了连接的和非连接的类别之间的关卡。我们接下来要深入考察一下这两种类别的不同之处。

1.3.1 连接对象

面向连接的部分是，那些在与数据源交互和处理时必须要有打开的可用连接的对象。在ADO.NET中面向连接的部分主要包括如下对象：

- Connection：这个对象可以建立一个与数据源的连接。依赖于实际所用的.NET数据提供程序，连接对象可以自动为你池化物理数据库连接。它们并不池化连接对象实例，了解这一点很重要，但是它们会试图循环使用物理数据库连接。连接对象的例子有

OleDbConnection，SqlConnection，OracleConnection等。这些将在第4章进一步讨论。

- Transaction：有时，你可能需要把一系列命令组织到一起或作为一个原子操作来执行，即“要么都执行，要么都不执行”。例如一个银行的应用程序，如果贷方不能执行操作，那么其借方也不应该执行操作。事务对象可以把一组命令组织起来原子化地执行。事务对象的例子有OleDbTransaction, SqlTransaction, OracleTransaction等。在ADO.NET 2.0中，你可以用System.Transactions命名空间来运行分布式事务，并支持非数据库事务。在ADO.NET 1.0和1.1中，使用System.EnterpriseServices命名空间可能并不是理想的解决方案。在第11章将会做个比较并讨论更多的细节。
- DataAdapter：这个对象可以作为ADO.NET中连接和非连接的网关。它可以建立一个连接，也可以提供一个已建立的连接。它拥有能够处理一个非连接对象的数据的足够信息，也可以基于事先没有指定的方式操作数据库。DataAdapter的例子有SqlDataAdapter，OracleDataAdapter等，将在第7章讨论它。
- Command：这个对象表示可在底层数据源上执行的命令。这个命令既可返回结果，也可不返回。这些命令可以用来操作、查询、更新或删除现存数据。此外，它们还可以用来操作底层数据表结构。命令对象的例子有SqlCommand，OracleCommand等。这些内容将在第5章讨论。
- Parameter：表示命令需要接收的参数。这就可以让命令更加灵活、接收输入值并且执行相应的操作。这些参数可以是存储过程的输入/输出值，或者是传递给SQL查询的“?”参数，或者是传递给动态查询的简单的命名参数。参数的例子有SqlParameter，OracleParameter等。这将在第5章讨论。
- DataReader：DataReader对象相当于是只读/只向前移的游标，可以用极快的速度从数据库获取数据，但是这种数据获取方式是只向前移和只读的。该对象将在第5章进一步讨论。

1.3.2　非连接对象

通常，只用连接对象并不能适应现代分布式应用程序的需求。非连接对象是使用ADO.NET构建的，只不过使用了一种不同的方法而已。非连接应用程序通常尽可能晚地连接，并且尽早地断开连接。在以非连接方式工作时，ADO.NET在不同的访问请求之间池化实际的物理连接。这将会在第4章讨论，我们会用实际的代码演示清晰地展示应用程序能通过这种连接池提高好几倍的性能。

ADO.NET非连接模型下需要用到如下对象：

- DataSet：DataSet是ADO.NET非连接数据访问模型的核心。理解DataSet最好的方式就是，把它看作是完全在内存中表示的非常小的关系型数据库管理系统（RDBMS）。但它并不完全是RDBMS，而且永远也不能取代RDBMS，它的各个组件与大多数主要的RDBMS对象一一对应连接起来，这能帮助你理解DataSet。同时，DataSet是通过System.Data.DataSet来提供的，意识到这一点很重要，即数据集位于任何.NET提供程序（provider）之上，这样就把数据集变成与.NET数据提供程序无关（下一节将更多地讨论.NET数据提

供程序)。DataSet也可以看作是DataTable和DataRelation的逻辑集合。

- DataTable：DataTable非常类似于数据库中的表。它是由DataColumn、DataRow以及作用于它们上的各种约束组成的。它以行/列格式存储数据。从ADO.NET 2.0开始，DataTable可以完全转换成XML，而且可以像DataSet那样被序列化。为满足数据访问的需要，可能在DataSet中就只包含一个DataTable，此时使用DataTable可能会更好。在以后的章节中你会看到，这样做不仅更方便，而且性能也更高。
- DataRow：DataTable的一个属性是DataRowCollection类型的Row，它表示一个可列举的DataRow对象集合。在数据填充到DataTable时，DataRowCollection获取一个新的DataRow对象，并且添加到其自身。在数据库中与DataRow最为贴切的逻辑对应就是数据表中的行。
- DataColumn：DataTable也包含一个DataColumnCollection类型的Column属性。本质上，这表示一个DataTable结构。在数据库中与DataColumn对象最为贴切的逻辑对应就是数据库中给定数据表的单个列。
- DataView：DataView很像是数据库中的视图。DataView可以让你在一个DataTable上创建一个“视图”，并且可以基于其Filter属性指定的一个预先设定的条件以查看数据的子集。你也可以使用Sort属性来对DataTable过滤后的子集数据进行排序。一个DataTable上可以定义多个视图。
- Constraint：DataTable也包含一个名为Constraint的ConstraintsCollection类型的属性。这就让你可以创建ForeignKeyConstraint或UniqueCosntraint对象，并且能把各个列关联到特定的条件上，在DataTable中，数据必须符合这些条件才能保存到DataTable中。在数据库中与ForeignKeyConstraint最为贴切的逻辑对应是外键，而UniqueConstraint指定了数据库中给定列的Unique条件。
- DataRelation：DataSet就像数据库一样，可以包含多个相互关联的表。DataRelation对象可以让你指定不同数据表之间的关系，这样就能实现跨数据表之间的数据验证，并在多个DataTable之间浏览父行和子行的数据。在数据库中最为贴切的逻辑对应是两个被指定的数据表之间的外键。ForeignKeyConstraint和DataRelation之间的差异在于：DataRelation除了能验证数据，还能让你很方便地在DataSet中的父行和子行之间遍历。

图1-3显示了各种连接和非连接对象如何应用到一起。

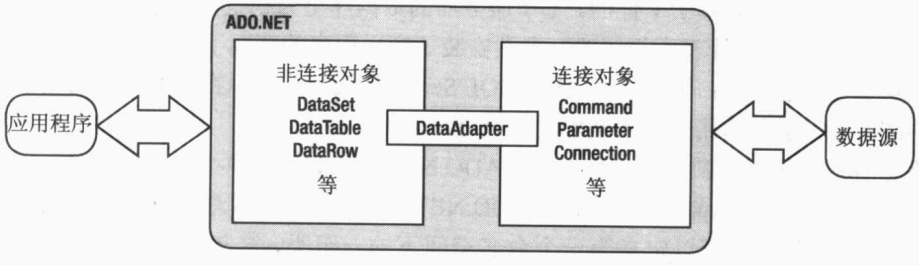


图1-3 ADO.NET的连接对象、非连接对象以及DataAdapter