.NET开发丛书

ADO.NET本质论
Essential ADO.NET

.NET开发人员心目中的好书！

（美） Bob Beauchemin 著
周 靖 译

清华大学出版社

.NET 开发丛书

ADO.NET 本质论

（美） Bob Beauchemin 著

周 靖 译

清华大学出版社

北 京

内 容 简 介

本书由资深数据库教师编写，全面介绍了 ADO.NET。书中深入剖析了 ADO.NET 的本质，探索了类、接口、属性和方法的工作原理，同时还为其他数据访问 API（包括 OLE DB，ADO，ODBC 和 JDBC）的程序员，提供了有价值的参考材料。本书适合具有一定数据库基础的开发人员阅读，也可用作数据库中高级课程或培训班配套教材。

北京市版权局著作权合同登记号 图字：01-2002-6533

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

图书在版编目（CIP）数据

ADO.NET 本质论/（美）布启敏著；周靖译.—北京：清华大学出版社，2003

（.NET 开发丛书）

书名原文：Essential ADO.NET

ISBN 7-302-07281-7

Ⅰ.A… Ⅱ.①布… ②周… Ⅲ.数据库—接口—程序设计 Ⅳ.TP311.11

中国版本图书馆 CIP 数据核字（2003）第 084068 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社总机：（010）62770175　　**客户服务**：（010）62776969

文稿编辑：文丹棋

封面设计：立日新设计公司

印 刷 者：北京四季青印刷厂

发 行 者：新华书店总店北京发行所

开　　本：185×260　**印 张**：23.75　**字 数**：470 千字

版　　次：2003 年 9 月第 1 版　2003 年 9 月第 1 次印刷

书　　号：ISBN 7-302-07281-7/TP・5286

印　　数：1~4000

定　　价：49.00 元

读者如是说

★★★★★ 正在进行 ADO. NET 编程的您，请不要错过！

毫无疑问，本书作者 Bob Beauchemin 是这方面的资深专家！从事数据库编程并打算学习ADO. NET的人都应该专心研读这本书。从书中可以看出，Bob 对数据库访问标准和ADO. NET具有很深的造诣，这是毋庸置疑的事实！在此之前，我读过很多涉及ADO. NET的书，但从精炼和清晰的角度而言，没有一本能与本书媲美。对于已具备 ODBC，OLE DB 和 ADO 基础的我，显然更喜欢阅读其中的常用概念比较，更偏爱其中指点我如何完成不依赖于其他标准来完成自己的工作。

★★★★★ 字里行间透露出作者具有丰富的经验和精深的专业知识！

本书最适合具有一定基础的数据库开发员。书中清楚地区分了连接和断开访问。后续章节为打算换用ADO. NET的其他数据库用户精心提供了“必读内容”。大多数代码虽然只是片段，但作者在本书配套网站上提供了完整的程序。

尽管本书最开始依照常规介绍了 SQL 及其相关模型，但大多数章节涉及较深主题，建议读者具备一定的基础。但这并不是说本书“门槛很高”，您只要稍微具备据库基础知识，就会发现本书包容了 ADO. NET 的所有内容。

本书可轻松阅读。各章均有代码片段，有的还提供了完整的程序。令我高兴的是，在亲手编译这些代码的过程中，我没有发现任何错误。第 2 章中的示范程序基于 SQL Server，但很容易将这些例子转换为另一个提供程序。下载的代码中还包含一些 Palm 数据库文件。第 4 章的代码片段可以到网上下载（不过，下载时需要仔细核对）。我下载的代码片段都能编译和执行，虽然不如完整的代码有用。

本书组织良好，内容丰富。作者显然具有丰富经验，而且学识渊博。

★★★★★ 好书，ADO. NET 用户必备之宝典！

Bob 的书详细地讲解了ADO. NET相关知识，他并没有纯粹用满篇的代码来“应付”读者，让读者自行理解，而是先提出一个概念，接着提供恰到好处的示范代码，确信读者完全理解之后，再介绍下一个概念。本书虽然是作者的“处女作”，但也说明作者内功深厚。本书可读性强，组织结构清晰，解决了大多数开发人员对于如何“炮制”代码的存疑。

坦白地说，本书不太适合新手阅读，这不是简单的“How To”，或“…21 天速成”，它的作用是抛砖引玉，引导您步入ADO. NET“殿堂”，而不是教您快速写代码。作者真正揭示了

ADO. NET的本质。

★★★★★ 一本不错的好书!

本书确实用通俗易懂的语言讲述了ADO. NET的工作原理。我觉得作者肯定是一个非常精通 SQL Server 的高手。

从头写一个严重依赖于数据库的应用程序需要非常周到的规划。读过本书之后,可"依葫芦画瓢"设计出一个有效的数据层,然后处理并发性、提供程序细节以及整体性能此类棘手的问题。

这是一本不错的书,期待着能早点看到它的新版本。

资料来源:*http://www.extremesqlserver.com/books/default.aspx? ASIN=0201758660*

译者序

由于工作需要，我对ADO. NET技术比较关注，同时也注意到国内ADO. NET方面的书乏善可陈，身边不少朋友表示自己迫切需要一本非常权威的ADO. NET宝典。好的ADO. NET书籍应该具备什么样的条件呢？在我看来，它至少有以下几点：第一，要全面而深入地剖析ADO. NET，不"放过"ADO. NET的每一个重要对象；第二，要对古往今来（有点夸张吧）的数据访问技术进行横向对比，让读者一眼就明白应该如何选择；第三，要知道读者需要什么，哪些应该重点强调，哪些应该点到为止；第四，要考虑到原有数据访问技术到ADO. NET的过渡；第五，向读者传达的信息必须是可靠的、经过证明的。

这几点，《ADO. NET本质论》都具备。首先，全书的开头和结尾（第1章和第10章）介绍了与数据有关的主题：模型、应用程序和API，第2～7章则深入剖析了ADO. NET的类、接口以及方法的工作原理（要想发挥ADO. NET的潜力，了解其幕后的工作原理是关键）；其次，书中对数据访问技术的过去和未来侃侃而谈，对ADO. NET、ADO、OLE DB以及其他数据访问API进行了对比，让读者一目了然；第三，本书的重点在于讲解工作原理，教您从更深的层次去思考，而不是教您一些机械、刻板的做法；第四，分别为OLE DB程序员、ADO程序员、ODBC程序员、JDBC程序员提供了迁移到ADO. NET的路径，这一部分内容包含在第9章；第五，从前言以及译者对书中示范代码的编译可以看出，本书的确为作者深思熟虑之后所写的，是一本经得起考验的书。

本书作者Bob Beauchemin从事信息处理已有20多年。他阅历丰富，不仅从事过软件开发、系统管理，还是一个活跃的撰稿人，在很多技术期刊和杂志发表过一些具有影响力的文章。我在一些专业类网站也时常看到他撰写的文章，他对数据库以及软件开发的精通程度令人叹服，简直就是一部"存储量"惊人的"参考书"！Bob学识渊博，博文广识，非常精通各种数据库。作为一个痴迷于技术的人，他常常不惜牺牲自己的业余时间，帮助他人排忧解难，先后帮助*Transportation COM +*和*Essential . NET*的作者解决他们写书过程中碰到的难题。这一点，我深有体会，在与Bob沟通的过程中，他总是来信必复，并把其中涉及到的前因后果分析得头头是道，使人茅塞顿开。

翻译本书虽然失去了部分休息时间，但通过跟Bob以及其他ADO. NET爱好者的沟通，不但使我受益匪浅，还使我进一步巩固了自己的数据库实践知识，有机会重新调整自己原有的数据库知识体系。

有的读者曾抱怨本书没有提供完整的示范代码，没错，书中大部分代码只是片段，但它们都能通过编译。如果真的需要完整的示范代码，也可径直前往本书配套网站下载，速度非常快。为方便读者，我已尽可能修正了原书中已有的错误（已经过作者确认），并将其合并到译文中。如果您在阅读过程中，还发现有"漏网之鱼"，欢迎您与作者本人或我联系。最后，希望本书成为您的良师益友，为您提供好的指引。

周靖于北京

原序一

写一本书真的很难。我自己的书就写了两年半，从1998年末，一直写到2001年初。整个过程就像一场噩梦。我的目的是为读者提供一本全面、权威的书，书中论及常见的、可伸缩的分布式系统设计。该书的常规主题是可伸缩的分布式系统设计，但重点集中于COM+的使用。没有Bob的帮助，完成该书对我而言，几乎是不可能的任务。

Bob是我的好参谋。在我的书快要收尾时（已经是第二年的事情了），我开始频繁地给他打电话和提问题。每次通话的内容大致相同。一般由我开头，没有半句客套话，直截了当地问："我做x的时候，却发生了y——错在哪里?!??"Bob会非常有耐心地回答："等一等，让我启动一下机器，运行z来看看……。"（在此期间，我开始相信Bob拥有迄今为止出现过的所有数据库，而且它们都在他家的一台机器上运行着。有时，我忍不住很想问他一些子虚乌有的数据库，比如QuuxBase，以证实自己的假设。但我最终放弃了这一想法，因为怕他居然有）。

Bob是一个价值不可估量的资源。例如，他帮助我理解了事务隔离在SQL Server中的工作原理（更重要的是在Oracle中的工作原理）；怎样使用OLE DB和ADO来执行批处理语句；OLE DB怎样实现连接池（以及如何用COM+的池化对象来更好地做这件事情）等等。

Bob曾经问我，他是否应该写一本有关OLE DB的书，而我的回答总是相同的："只要您觉得可行，就可以放手去写。"等Bob准备好写作时，OLE DB（和COM的其他东西一样）开始逐渐被ADO.NET取代，所以他的重点马上投向了这个新的、充满挑战的领域，虽然这并不算是一个全新的领域。

据我所知，Bob对于过去与现在的数据库及数据访问技术的精通程度无人能及。只有他才有能力将ADO.NET跟ADO、OLE DB以及您所知道的任何数据访问API之间的异同进行深刻而全面的对比。随着.NET框架中的类逐渐取代您用过的所有API，这一同过去的联系将帮助您理解新的Microsoft平台的妙处。

无论您编写数据访问使用程序还是编写提供程序，无论使用ADO、OLE DB还是或新或旧的API，都可参考本书。

Tim Ewald

*Transactional COM +*作者

原序二

我们的生活中，几乎每一方面都由某个数据库来控制。我们所做的一切都被分类，并持久地保存下来，然后通过分析，以邮寄广告或者一次促销活动的形式，返回给我们。如果您开发过软件，可能会在自己的方案中采用一个数据库。数据库固然重要，但深入了解如何有效地访问它们以及如何处理其中的数据，更不容忽视。数据具有不同的形式和大小，有许多不同的方式来存储数据。如果您如果您有幸只用一个数据库，那么肯定非常精通那个数据库的数据访问 API。但是，很少有方案只用一个数据源，即使现在只用一个，也不能保证以后总是使用那一个。在方案的升级版本中，您的客户可能选择其他厂商的数据库。

统一数据库访问一直是库提供者奋斗的目标，有了它，便可以使用统一 API 来编写相同的代码，无论使用的是什么数据源。过去几年出现了几种统一 API。其中，Microsoft 的 API 已经以 ADO. NET 的形式稳定下来，ADO. NET 本身也成为 . NET 框架的一部分。. NET 战略代表着用运行库和垃圾回收机制进行的一种全新的编程方式，而 Microsoft 数据库团队利用这个机会将 XML 集成到 API 中。XML 不仅是一种数据源，还是一种统一数据格式和一种数据库通用语言。ADO. NET 是一个巨大的库，需要全面的讲述。

“全面”和“权威”这两个词精辟地概括了 Bob Beauchemin 所著的《ADO. NET 本质论》。从第一页翻到最后一页，您会一直沉浸在 ADO. NET 博大精深的世界中，您会跟随着作者细致的讲述，深入了解库的工作原理，学习如何将其用于自己的代码中。第一次阅读时，可采取“快读”的方式快速从头翻到尾。只有在遇到示范代码时，才应该暂停一下，亲自录入这些代码，并在可能的情况下，仔细分析这些代码。这样可以实际体验 ADO. NET 的功能，以及具体如何访问这些功能。经过这一次快读之后，您应该能有效地开发数据库方案，但以后编写自己的代码时，仍需反复精读某个主题。Bob Beauchemin 是一个不折不扣的数据库大师，Bob 的任何书都是数据库开发人员必读的，您能获得对一种技术的最好的解释。通过本书的学习，您不仅知道如何使用 ADO. NET，还能理解库的基本设计原理。当然，也能从中了解 . NET 许多鲜为人知的奥秘。

Richard Grimes

Developing Applications with Visual Studio . NET 作者

前　言

本书将讨论ADO. NET，这是一种新的、基于提供程序模型（Provider Model）的数据访问库，它是 Microsoft. NET 战略的一个组成部分。但本书同时还将讨论数据——在什么地方存储，以及如何检索、更新及索引数据。由于要讲述数据管理的细节，所以本书的覆盖面有点儿大，不会只是简单地罗列所有类、接口、属性以及方法在一个库中的用法。您必须仔细研究准备用这个库来解决的问题，确定要使用什么“应用程序编程接口”（API）或者使用 API 的哪一部分。

虽然商业问题都可划分为特定的、重复出现的模式，但不可能有一成不变的规则集成了像菜谱那样的“傻瓜”方案。涉及数据访问、表示和交换时，这一问题尤为突出。今天为了追求终极速度而采用原生汇编语言来编程，明天就会成为维护时的噩梦。对某个用户来说，一种完美的工作模式在一组用户中却无法施展。适用于一个小型工作组的模式在应用于整个企业时，却显得困难重重。虽然本书确实为一些泛化的问题提供了解决方案，但肯定没有为您提供任何 C&P（拷贝和粘贴）方案。

泛化和专用编程模型

不久以前，我的观念一直是坚持使用厂商专用的、屡经考验的编程接口，因为它们通常总是最快的。但是，在我的职业生涯中，必须经常进行应用程序的转换，因为一个企业可能决定改变硬件类型或者数据库。在任何有悠久历史的公司中，就像时装的换季那样，硬件和数据库的改换是常有的事。

目睹一家公司的网站准备从厂商专用数据库 API 转换成开放数据库连接（Open Database Connectivity，ODBC，一种更抽象的 API）的时候，我的观念第一次发生了变化。这个工作让我体会到了一个“抽象的、提供程序样式的接口”的巨大好处。对于许多公司来说，实际选择的数据库要取决于公司其他部门的意向，这不受我们项目的控制。我的项目领导人当时决定选择 ODBC，而不是选择原生 API。事实很快便证明了这个决定是正确的。新的应用程序很快就写好了，数据库最终也生成了。项目按期完成，项目领导人正好赶得上周五下午的会议。周一早上，我就得到通知，他们最终选用的数据库发生了改变。幸好，因为我们使用的是 ODBC（同时因为这两种数据库都有 ODBC 驱动程序），所以我们只需生成新数据库，对程序进行少量修改（因为两者的 SQL 格式有细微差别），重新测试程序，就完成了应用程序的转换。我们太幸运了。

基于提供程序的数据访问 API 必须平衡两方面的问题：抽象出泛化的功能；同时为厂商提供足够大的自由发挥的空间，确保 API 不会成为一个难以使用、性能极差的怪物。

ADO.NET在后一个方面做得很好：SQL Server 专用的数据提供程序（称为 SqlClient）证明 Microsoft 的厂商专用扩展做得非常出色。另外，将来问世的数据提供程序还将证明 Microsoft 的更多过人之处。

技术轮回与革新

我从事计算机信息处理工作已有多年历史。接受现在这份工作（在 DevelopMentor）的面试时，他们问我："您的年纪逐渐增大，编程语言、设计模式和应用程序编程接口又不断发生变化，您有没有觉得掌握新技术会越来越困难？"我微笑着回答说："这实际会变得更容易"。虽然计算工业经历了几代"新思维"的转变，但我发现被抽象的"问题域"本身却没有改变。每次进行抽象时，抽象本身的差异通常都不大。除此之外，抽象有时完全就是一种轮回；今天的"新思维"，说不定就是新瓶里面装的旧酒。

例如，我对数据访问和数据库管理系统产品的结构一直抱有浓厚的兴趣。我以前将文件用作数据库（或至少算是数据存储），使用过层次化数据库，使用过 CODASYL 数据库（由 Conference on Data Systems Languages 标准化，CDSL 是美国国防部下属的一个组织，主要任务是发展程序语言，现在已经不复存在。——译者注）、关系数据库、对象数据库、多维数据库以及它们的所有变体。第一次接触作为数据模型使用的 XML 时，我同样对它产生了浓厚的兴趣。它的数据抽象模型、它的驻留内存对象模型、它的丰富的数据处理工具以及它的查询语言……，所有这一切给我留下了深刻的印象，但我对它有一种"似曾相识"的感觉。后来，我找到一本对关系和层次化数据库进行比较的旧书，立即就发现 IMS（IBM 在 20 世纪 80 年代中期以前的旗舰数据库）中的层次化导航函数与 XML 文档对象模型（DOM）API 中的那些函数极为相似。我后来还发现，XPath 查询语言采用的思路和当时一样：让基础 SQL 引擎来完成导航功能。以后要提高速度，只需对查询引擎本身进行改进，而不必重写基于导航的程序。

另外，目前在不相似的系统之间进行商家到商家的数据交换时，通常使用一种已知的数据格式（XML）。这种模式类似于我以前用过的一个系统，它用著名的"自动票据交换"（Automatic Clearinghouse，ACH）格式来导入和导出银行业务。它使用一种已知的数据格式和远程通信协议在不相似的系统之间交换数据。

能在一种新技术中找到某些熟悉的东西，肯定是有用的。但我发现，当我谈论一种新技术时，如果老是说："它就像……"，我的同事们就会感觉十分不爽。他们不爽是因为我好像在说这种技术中没有什么新东西，没有什么改进。但我真正的意思与此相反，而是说它们通常都会进行一些改进，这些改进基于要解决的新问题，而且可能会以一种新的方式来看待现有的问题。对我而言，"它就像……"代表着从旧世界到新世界的一座桥梁，而我就根据这两个世界的联系来看待所有的改进。我发现，当我提到新技术上的改进时，同事们就显得比较高兴了。

我在 DevelopMentor 的主要工作之一就是设计和讲授 OLE DB 课程。在我四处旅行授课的过程中，学生们会问："它和 ODBC 有什么不同？"或者"我为什么要从 JDBC 转变到它？"

总而言之，他们的问题就是：我得到了什么，我失去了什么，以及两者有何相同之处？在多次回答这些问题，同时研究了许多技术所发生的改变之后，我突然发现自己已经完全有资格对新的ADO.NET接口的影响进行评定。

顺便提一句，作为JDBC的发明者，Sun Microsystems公司声称JDBC不是任何东西的缩写形式。许多人以为它的全称是Java Database Connectivity，但Sun否认了这一点，虽然该公司自己的部分文档中仍在沿用这样的说法。

数据访问的改变

讨论数据访问和表示策略时，我习惯于开玩笑地说，Web浏览器不过是一种“可以唱歌和跳舞的彩色终端”。但是，借用一段陈词滥调：通过Web浏览器来访问应用程序全面改变了我们设计数据访问策略的方式。本节将对此进行解释。

首先，用户数量可能非常多，而且频繁变化。在终端机的时代，用户数量是可以准确预计的，或至少是可以控制的。要增大系统的规模，可在私有网络中增添更多的终端机和用户。用户还必须先填表才能访问公司的应用程序，所以我们能对用户群体的增长进行控制。但是，在Internet时代，您的应用程序的用户数量只受限于这个应用程序的流行度。这正是Internet作为一种商业通信工具的最大的卖点。公司希望从一台小的服务器开始，并逐渐扩充到更大的用户群体。当然，在这个过程中，公司只能获得有限的预见性。它改变了我们设计和编写应用程序的方式。

其次，用户不会“注销”Internet应用程序，您有时根本不能通过物理方式识别他们。Internet应用程序中的超链接允许用户立即跳转出去，以检查自己喜爱的股票或者球队。如果其他网站包含了新的和有趣的东西，他们也许永远不会回到您的应用程序。因此，不可能在Web应用程序中集成注销逻辑，也不能保证它的使用。另外，由于人们普遍使用动态地址分配协议（比如DHCP）和代理服务器软件，所以您也不可能根据终端ID来识别用户。结合用户不会注销应用程序这一事实，您不可能像在终端时代那样，根据一台计算设备的位置，为每个特定的用户都保留一定的存储空间。这是另一个巨大的改变。

最后，用户自己完成大量数据输入工作。在旧式系统中，悲观并发性和记录锁定之所以能够正常进行，是因为数据输入由一组终端操作员来完成。如果两个操作员同时访问与您的账号有关的数据，而且您的数据正在“忙”，那么其中一名操作员可能暂时放下您的记录，等待以后录入。Web应用程序则不同，它将每一个人都变成了数据录入员。由于不大可能有“两个您”同时更新您的个人和财务信息，所以乐观并发性显得更实用。它还意味着数据编辑技术和业务规则必须变得更可靠，因为非专业的录入员可能录入乱七八糟的东西。

ADO.NET API采用的数据访问模式根据今天的数据访问和输入现状进行了调整。解决今天所面临的问题时，连接池、断开式更新以及乐观并发性是关键——即使是在使用原有的技术时。

本书的结构

第 1 章“数据:模型、应用程序和 API”要反映的主题是,一个数据访问 API 必须在两个方面取得某种平衡:一方面坚持采用一种数据存储样式或者数据库;另一方面封装尽可能多的功能,因为单独的类和方法是没有意义的。本章列举了ADO.NET中的新功能,并描述了它最适合用于哪些数据存储样式和应用程序类型。

第 2 章“ADO.NET基础”分 3 部分来讨论ADO.NET:数据提供程序、ADO.NET DataSet 对象模型以及 XML 数据访问模型。每个部分都通过一个简单的程序来阐明。XML API 以及 XML 与ADO.NET的集成问题将在第 7 章进行讨论,因为我认为 XML 堆栈是.NET框架的总体数据访问堆栈的一个重要组成部分。

第 3 章“连接模型:流式数据访问”指出,大多数数据访问 API 都以一个提供程序模型为中心,这个模型对各种具体实现之间的细微区别进行了抽象。本章解释了ADO.NET对于您熟悉的、用于访问关系数据存储的“连接、命令和结果”模型进行的修改。本章最后总结了一些应用程序特有的特性,包括连接池和自动分布式处理。

在第 4 章“DataSet 类:关系数据的集合”中,您会了解到大多数数据访问 API 都包括“驻留内存的数据表示”的概念,可利用它在窗体上填充控件,并可一边遍历数据,一边进行处理。在ADO.NET中,这个模型基于 DataSet 类及其为它提供支持的集合类,它们搭建起了一个关系数据库。本章在介绍 DataSet 时,重点讲解了它作为一个独立的对象模型的复杂性。

虽然 DataSet 是在内存中表示数据的一种便利方式,但它最大的用处还是作为一个客户端的数据库数据模型来使用。第 5 章“DataAdapter:数据库和 DataSet 的同步”讲解如何将数据从一个数据提供程序传输到一个 DataSet,并将用户以脱机(离线)方式对 DataSet 进行的更改存回数据库,使这种更改持久化。由于数据库供许多用户共享,而这些用户有可能同时操纵相同的数据,所以本章讨论了多个用户在脱机模式中更改相同的数据时要解决的问题,这种问题称为“乐观并发性冲突解析问题”。除此之外,关于在什么情况下使用一个“直接来自数据库”的流(ADO.NET DataReader),以及在什么情况下使用一个脱机缓存(ADO.NET DataSet),并配合通用的应用程序访问模式,人们还持有不同意见。本章对此提供了一些指导原则。

如果您想通过一个图形化程序或者网页来表示数据,就必须将数据库或数据缓存中的数据映射到一些图形控件,比如文本框、列表框和数据网格等。第 6 章“数据绑定:ADO.NET 和 GUI”讲解了 ADO.NET 与用户界面元素的集成。本章再次讨论了 DataReader 和 DataSet,不过这一次是在使用ASP.NET Web 窗体表示风格的前提下。

由于世界上的大多数数据都存储在关系数据库中,所以第 7 章“XML 和数据访问的集成”探讨了ADO.NET中用于对“关系数据存储”和“XML 表示/处理”进行集成的各种方法。改进的 XML 集成能力是ADO.NET的主要优势之一。本章最后总结了 XML 与 Microsoft 旗舰数据库 SQL Server 进行集成的问题。

数据访问(以及其他模型,前提是它们具有多个类似的实现,只是在细节上有所区别)是通过一个提供程序模型来抽象的。第 8 章“提供程序:ADO. NET和数据提供程序”探讨了“OLE DB 提供程序模型”到“ADO. NET数据提供程序模型”的映射关系。另外还解释 XML Infoset 提供程序的概念,它是基于 XML Infoset 模型的、对提供程序模型的一种抽象。第 9 章“消费者的ADO. NET迁移路径”为数据消费者(比如编写数据访问代码的程序员和使用数据的程序,译文一般保留 consumer 原文。——译者注)提供了一个有用的参考,这些 consumer 可能是从其他数据访问 API 迁移过来的,比如 ADO、ODBC 和 JDBC 等。本章的宗旨是让具有不同背景的人快速熟悉ADO. NET,能将他们的 X 类或者 Y 函数映射到 ADO. NET的类和方法。

本书第 1 章就概述了各种类型的数据存储以及使用这些数据应用程序。作为本书的最后一章,第 10 章“ADO. NET和各种数据类型”将根据您已经学到的知识,再次讨论第 1 章提出的一些问题,探讨数据堆栈(ADO. NET和 XML)是否真的包含一个适用于每个人的方案。最后,对数据访问的未来进行了展望。

如何阅读本书

本书分成 3 部分:陈述和分析、通过实例提供的 API 参考,以及为需要进行数据转换的程序员提供的纯粹参考性的材料。理想情况下,您应该顺序阅读,但首次阅读时,没有必要完全理解其中的内容。在阅读过程中遇到难点,比如“我为什么要关心这个?”或者“我对此没有概念”,不要停顿,继续读下去,通过阅读后文,难以理解的东西会逐渐明朗化。

第 1 章和第 10 章从广义上陈述和分析了数据库类型、数据库 API 以及应用程序风格。您不一定完全理解或者同意其中的每一个论点。第 2～7 章详细解释了ADO. NET类、接口和方法的工作原理。您既可以全部阅读,也可以选自己感兴趣的阅读,但第一次最好按顺序阅读。第 8 章和第 9 章分别是为 provider 和 consumer 的作者准备的,他们可能正在使用其他库,并希望寻找一个迁移到ADO. NET的途径。这两章属于参考资料。对于那些还没有写过 OLE DB 代码的人,第 8 章的内容可能有点难以理解。本章是为我的一些朋友和学生而写的,过去几年里,我和他们一起学习和编写过 OLE DB 提供程序,而且他们总是抱怨没有人为他们专门写一本像样的书。

本书没有花费几百页的篇幅来罗列各种类、接口、方法和属性信息。这方面的信息最好参考. NET框架软件开发工具箱(SDK)的联机文档。花了几小时时间浏览 Windows 平台文档后,我对 Windows 平台就已经非常熟悉了。不仅框架 SDK 文档做得不错,OLE DB 程序员的参考文档也给人留下了深刻的印象。

观察与文档

本书之所以没有早点儿出版,是由于几方面的原因。一个是我知道. NET产品在 Beta

阶段会发生变化，而且可能是非常重大的变化。我可不想在目前大量的“Beta 测试版编程”书中再增加一本，虽然这些书也有一定的价值。这种书在出版后的几个月之后，就有可能过时，并可能误导读者。

除此之外，我想通过直接试验代码的方式来验证文档中所说的东西。另外，我还尝试描述提供程序的一些内部行为，以及它们的实现细节。它们在 Beta 测试过程有可能变动（有时甚至是重大的变动），而我必须重写一些小节，并重新思考解决方案。我必须等待最终版本发布，这样才能放心运行代码片断，并再次进行试验。

仅仅通过试验和观察来推导内部工作原理是要冒一定风险的，因为这些隐藏细节的实现会随着时间的推移而发生改变。改变的原因可能是软件厂商不断更新和改进它们的产品。正是由于存在这样的风险，所以我尝试在本书中对我的试验进行文档化，使您能在软件或基础数据库发生改变之后，能够重复我的试验。

其他额外的代码、勘误（本来的错误，或者由于产品更新而产生的错误）以及其他额外的信息将发布于本书配套网站 *staff.develop.com/bobb/adonetbook*（本书简体中文译本已根据现有的勘误（截止 2003 年 7 月 4 日）进行了更新。——译者注）。

目　录