PROGRAMMER TO PROGRAMMER™

Beginning ASP.NET Databases using C#

ASP.NET 数据库入门经典

—C# 编程篇

John Kauffman
Brian Matsik 等著
张哲峰 黄翔宇 译

清华大学出版社
http://www.tup.com.cn

ASP.NET 数据库入门经典

——C#编程篇

John Kauffman
Brian Matsik
等著

张哲峰　黄翔宇
译

清华大学出版社

(京) 新登字 158 号
北京市版权局著作权合同登记号：01-2002-3206

内 容 简 介

对于 Web 站点而言，最基本的要求是能够动态地更新内容，能够准确、安全地传输和存储数据，这些都需要访问数据库。本书全面详尽地介绍了如何使用 C#和 ASP.NET 来开发数据驱动的 Web 应用程序。本书首先概述了数据驱动的 Web 站点的基本概念，以及开发环境。接着介绍了一些创建数据库的理论和 SQL 语言，对数据库的连接以及其他主题，如连接字符串的存储。之后，讨论了 DataReader 和 DataSet 对象，讲述了如何添加、更新和删除记录。然后运用 ADO.NET 的知识，讨论组件、应用程序和存储过程的创建，以及一些性能问题。最后提供了一个实例，将这些知识都应用到一个实际的应用程序中。

本书适用于具有一些 C#和 ASP.NET 编程经验的开发人员；熟悉.NET Framework 的开发人员；以及广大想要学习如何使用 ASP.NET 编写以数据为核心的 Web 应用程序的读者。

John Kauffman，Brian Matsik: Beginning ASP.NET Databases using C#
EISBN：1-86100-741-8

图书在版编目(CIP)数据

ASP.NET 数据库入门经典—— C#编程篇/(美)考夫曼等著；张哲峰，黄翔宇译.—北京：清华大学出版社，2002
书名原文：Beginning ASP.NET Databases using C#
ISBN 7-302-06144-0

Ⅰ.A… Ⅱ. ①考…②张…③黄… Ⅲ. ①主页制作—程序设计 ②C 语言—程序设计 Ⅳ.TP393.092

中国版本图书馆 CIP 数据核字(2002)第 097388 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.com.cn
责任编辑：于平
印 刷 者：北京牛山世兴印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：25.75　**字数**：659 千字
版　　次：2003 年 1 月第 1 版　2003 年 1 月第 1 次印刷
书　　号：ISBN 7-302-06144-0/TP・3675
印　　数：0001～5000
定　　价：48.00 元

出版者的话

随着国际互联网的快速崛起和迅猛发展，计算机之间的互联需求越来越迫切，而目前计算机硬件设备的不兼容性严重束缚了互联网的发展，引发了新一轮的跨平台软件的开发浪潮。软件商纷纷推出新的战略规划和解决方案，Microsoft 提出的.NET 战略就是其中的经典之作。

在经历这场浪潮的洗涤和考验过程中，全球的软件开发人员都迫切需要了解新的软件技术和开发思路。为了满足国内 IT 从业人员的需要，清华大学出版社从 Wrox 出版公司引进了若干套编程系列丛书，“入门经典”系列是其中不可或缺的入门之作。作为世界著名的编程技术图书的出版公司，Wrox 推出的这套“入门经典”系列丛书主要面向编程的初学者、需要了解.NET 策略的程序员，以及需要迅速掌握多门编程工具的程序员。该丛书依旧秉承了 Wrox 公司“由程序员为程序员而著（Programmer to Programmer）”的创作理念，每本书均由世界顶级的编程高手执笔。他们站在资深程序员的高度，循序渐进地为初学者讲述了.NET 的理念和构架、编程基本思想、编程语言基础、程序的控制和调试、Windows 应用程序的开发、对象编程技术、数据库访问技术、Web 程序开发和.NET 构架应用等最新的软件开发知识，同时辅以大量操作性强的程序为示例，为读者提供了清晰的编程思路和宝贵的编程经验。

为了保证该系列丛书的质量，清华大学出版社迅速组织了一批位于 IT 开发领域前沿的专家学者进行翻译，经过编辑人员的进一步加工整理后，现陆续奉献给广大读者。

读者可以从 www.wrox.com 网站下载所需的源代码和获取相关的技术支持。同时，也欢迎广大读者参与 p2p.wrox.com 网站上的在线讨论，与世界各地的编程人员交流读书感受和编程体验。

前　言

对任何 Web 站点而言，最重要的要求就是能够准确而安全地发送和存储信息。这种信息可以是任何形式，包括信贷公司提供的信用卡资料到市场信息网站的民意投票结果。不管将 ASP.NET Web 页面用于何种目的，您迟早都会需要处理涉及数据库的访问、读取和写入数据以及一般的控制操作。

幸运的是，操纵 Web 应用程序的数据库比以前更容易。.NET Framework 改进了开发复杂的 Web 站点的方式。ASP.NET 对如何开发复杂和交互的 Web 站点作了重大的改进，ADO.NET 提供了强大而灵活的数据访问功能，从根本上改变了从 Web 应用程序中检索、处理和存储数据的方式。

本书将介绍如何创建能够支持数据的 ASP.NET 应用程序。本书根据一些实践经验进行编写，并列举了演示基础原理的代码示例。本书要求读者对 ASP.NET 和 C#有一些基本的认识，如果对操纵数据库没有任何经验也没有关系，作者将指导每个过程，包括将 ASP.NET 页面与数据库连接，研究各种读取、处理和更新数据的方法。接下来研究所有涉及侧重于数据的 ASP.NET 编程的问题，包括一些高级的主题，例如使用存储过程和组件。本书还给出了案例分析，将前面几章讨论的内容都置于同一个环境中加以运用。

0.1　本书主要内容

第 1 章介绍创建数据驱动的 Web 站点的一般性概念。特别讨论了 Web 站点与数据之间的关系，在此过程中介绍了 ADO.NET。在这一章的末尾，还为本书其余章节创建了一个开发环境，讨论了一些可能会遇到的问题。

到底什么是数据库呢？这似乎是个奇怪的问题。第 2 章介绍了不同类型的数据库，如何设计它们，以及如何使用它们存储和检索信息。接下来又概述了 SQL 语言，并与 Microsoft 的数据库方案作了一个简单的比较。

了解了数据库以后，就需要学习如何将它们与 ASP.NET 代码连接。第 3 章首先概述了第 1 章提到的 ADO.NET，阐明了需要掌握的重要术语。接着是创建和测试对 Northwind 示例数据库的连接，本书将通篇使用该数据库。本章演示了大量例子，演示如何连接不同的数据源，例如 Access，SQL Server，Excel 和 XML，以便在任何开发环境中都游刃有余。

第 4 章通过一些例子，运用 DataReader 对象读取和显示数据。首先讨论了在 ASP.NET 和 ADO.NET 中处理数据的原理，在 ADO.NET 中引入了 Command 对象。然后介绍如何运用 DataReader 获取数据，接着将那些数据绑定到像单选按钮、复选框这样的不同控件上和最重要的 DataGrid 上。

第 5 章讨论了一个重要的 ADO.NET 对象：DataSet，它用于存储和检索服务器上任何复杂

形式的数据。在介绍完原理之后，使用大量的例子来演示使用 DataSet 的不同方法。

当然，我们并不总是读取其他人的数据，因此第 6 章介绍了如何在现有的数据库中创建自己的记录。该章的主题包括 ASP.NET 的验证控件，以及使用 Command 对象和 DataSet 对象实现插入记录。

第 7 章演示了如何修改和删除记录。介绍了如何使用 DataGrid 和 DataView 控件编辑数据，如何从 DataSet 对象中更新和删除记录。

第 8 章首先介绍了存储过程的概念，用途以及它们的一些优缺点。在演示完如何用 C#创建存储过程后，这一章又讲述了如何从 ASP.NET 代码中调用它们，如何提供一些参数去执行特定的任务。

第 9 章介绍了一些更高级的主题，这在创建商业的数据驱动 ASP.NET 应用程序时会碰到。这些主题包括错误处理、代码结构、可伸缩性，以及并发和安全性。

第 10 章介绍了组件，在讨论如何创建它们之前说明了它们的用途。该章中的示例代码主要是创建一个类库，包括访问数据库的大量方法。

任何编程人员的目标之一就是尽可能编写最好的代码，第 11 章讨论了性能问题。介绍了连接池和性能计数，对 DataSet 和 DataReader 对象在不同情况下的相对性能作一些衡量。还重点突出了尽可能地将旧的 ADO 代码移植为新的 ADO.NET 代码。

第 12 章介绍了一个案例分析。它以本书前面介绍的所有内容为基础，构建了一个完整的应用程序，使用了多个 ASPX 页面，ASP.NET Web 服务器控件，C#组件和存储过程。

0.2 本书读者对象

本书的读者应当具备以下条件：

- 有一定的 C#和 ASP.NET 编程经验
- 熟悉.NET Framework 和相关技术
- 想要学习如何使用 ASP.NET 创建具有数据处理能力的 Web 应用程序

读者并不需要知道任何有关数据库的知识，就可以学习本书。当然，具备一些入门知识总归要好一些。

0.3 学习本书所需要的前提条件

为了能够运行本书中的范例，您需要预先在您的 PC 上安装如下软件：

- Microsoft Windows NT、Windows 2000 或 Windows XP
- .NET Framework
- Visual Studio .NET 中的一个版本，至少要有 C#的标准版

另外，书中有一些使用 Microsoft Access 和 Microsoft SQL Server 的例子，但是它们对读者没有运行环境的要求。所有重要的例子都使用 MSDE(微软开发引擎，它随同 Microsoft Visual Studio .NET 的各个版本一起提供)作为它们的数据库引擎。

0.4 客户支持

我们一贯重视听取读者的意见，特别希望收到您对本书的反馈，例如，您喜欢什么，不喜欢什么，您认为我们下次如何才能做得更好。您可以发电子邮件到 feedback@wrox.com。请您在来信中务必写清楚书名。

0.4.1 如何下载本书的范例代码

请您登录 Wrox 公司的站点(地址为 http://www.wrox.com/)，只需通过使用 Search 工具或使用书名列表，就可以方便地定位需要的书目。然后，单击 Code 列中的 Download 链接，或者单击本书具体页面中的 Download Code 链接，就可以下载相应的范例代码。

从我们的站点上下载的文件都是使用 WinZip 压缩过的文档。将文件保存到本地磁盘上后，需要使用一个解压程序(例如 WinZip 或 PKUnzip)来解压文件。在解压文件时，通常将代码解压到每一章所在的文件夹中。在解压的过程中，应确保将解压程序(WinZip、PKUnzip, 其他)设置为使用的文件夹名。

0.4.2 勘误表

我们已经尽最大努力确保本书中的文本和代码没有错误，但是错误仍然在所难免。如果您发现本书存在错误，例如拼写错误或不正确的代码片段，请给我们发来反馈信息，我们将不胜感激。勘误表的发送可以节约其他读者学习本书的时间，而且能够帮助我们提供更高质量的信息。请将您的反馈信息发电子邮件到 support@wrox.com。您的反馈信息将被确认，如果正确，将被粘贴到本书的勘误页面上，或者在本书的后续版本中使用。

要在我们的站点上找到勘误表，请访问 http://www.wrox.com/，并通过 Advanced Search 或者书名列表定位本书页面。然后，单击 Book Errata 超链接即可，该链接位于本书的具体页面中的封面图形下面。

0.4.3 E-mail 支持

如果您希望直接向了解本书细节的专家咨询书中的问题，可以发电子邮件到 support@wrox.com，要求在邮件的主题中带有本书的书名和 ISBN 编码的后 4 位数字。一封典型的电子邮件应包括以下内容：

- 在主题部分中必须有本书的书名、ISBN 的后 4 位数字(7418)和出现问题的页数。
- 在信的正文部分应包括读者的名字、联系方式和问题。

我们不会返回给您无用的邮件。为了节约大家的时间，我们需要了解一些具体细节。当您发出一封电子邮件信息后，它将经过下面一系列的处理：

- 客户支持：首先，您的信息将被递送到我们的用户支持人员手中，他们是第一批读者。它们手中有最为常见的询问问题文档资料，能够立即回答有关本书或者 Web 站点的任何常见问题。
- 编辑处理：接着，一些有深度的问题将被送到对本书负责的技术编辑手中，他们在程序

设计语言或者特定的产品上有着丰富的经验，能够回答相关主题的详细技术问题。

- 作者处理：最后，如果编辑不能回答您的问题(这种情况很少发生)，他们将向本书的作者寻求答案。我们将尽量保护作者免受干扰，以便不影响其写作。然而，我们也非常高兴转寄给他们一些特殊的问题。所有 Wrox 公司的作者都为他们的书提供技术支持，他们将发电子邮件给用户和编辑，进而使所有的读者受益。

Wrox 公司的支持部门仅仅对那些与我们出版的书目内容直接相关的问题提供支持，对于超出常用书目支持的问题，您可以从 http://p2p.wrox.com 论坛中的公共列表中获得支持信息。

0.4.4 p2p.wrox.com

如果想和作者及同行进行讨论，请加入到 P2P 邮件列表，而且我们独特的系统除了提供了一对一的邮件支持系统以外，还通过邮件列表、论坛、新闻组等联系方式进一步体现了 programmer to programmer™(由程序员为程序员而作)的编著理念。如果您向 P2P 发送一个问题，请放心它一定会得到登录邮件列表的 Wrox 公司作者和其他相关专家的帮助。无论您是在阅读本书，还是在开发自己的应用程序，都可以在 p2p.wrox.com 站点中找到许多对自己有所帮助的邮件列表。针对本书的学习，最合适的列表是 aspx_beginners 和 c_sharp。

要想订阅一个邮件列表，需要完成以下的步骤：

(1) 登录 http://p2p.wrox.com/站点。

(2) 从左边的主菜单栏选择适当的类别。

(3) 单击希望加入的邮件列表。

(4) 按照订阅说明填写自己的邮件地址和密码。

(5) 回复您收到的确认邮件。

(6) 使用订阅管理工具加入更多的邮件列表并设置自己的首选邮件。

0.4.5 本系统为什么能够为读者提供最好的客户支持

您可以连接到整个邮件列表，也可以只接收每周的邮件摘要。如果您没有时间和工具来接收邮件列表，可以直接查找我们的在线文档。独特的 Lyris 系统可以将一些没用的垃圾邮件删除，并保护您的电子邮件地址不被侵扰。当有加入和离开列表的问题，以及任何有关列表的其他常见问题时，请将邮件发送到 listsupport@p2p.wrox.com。

目　　录

第1章　在Web上显示数据

当 Web 首次出现时，人们必须用一个比喻来说明如何在 Web 上显示信息。如果看一下那个时期的 Web 站点的例子，其内容大部分都是在传统媒体上所能看到的内容，如书籍、杂志和报纸。这导致了 Web 与其他媒体的用途相同；它只是在创建页面的时候提供了同样的信息。当然，那样做并没有错，但它对 Web 的真正用途造成了限制。

随着时间的推移，支持 Web 的技术发展日渐成熟，Web 站点从只能提供静态内容转变为也能够提供动态的应用程序。这些应用程序允许用户选择他们感兴趣的信息，提供了定制的用户操作，可实时对它们进行修改。

这些应用程序的关键就是它们所包含的数据。不管什么内容——可能是一个产品目录、一组客户资料或是一个档案库——都是数据使得它们动态化的。在过去，通过 Web 提供数据比通过传统的桌面应用程序更困难，由于使用的开发工具和功能，以及 Web 本身特性的缘故，使得用户不得不放弃这类应用程序和数据。但随着时代的发展，特别是 Microsoft 的.NET Framework 的出现，这一情形得到了改观。现在，Web 应用程序开发人员可以与桌面应用程序开发人员平起平坐了。

本章将详细介绍数据驱动的 Web 站点，以及如何在 ASP.NET 中实现它们。首先是讨论数据驱动站点的优缺点，接着提供这类数据的数据源。在这之后学习.NET Framework 的数据访问技术——ADO.NET——包括它的体系结构、它的类以及它是如何适应数据驱动应用程序的结构。最后介绍本书将一直使用的数据库服务器的安装。

说明：

数据驱动的 Web 应用程序是显示动态数据的 Web 站点。用户的经验能够改变对保存在数据存储器中信息的反映。

1.1　数据驱动 Web 站点的优缺点

数据驱动系统的一些优点很明显，但也有一些不是很明显的优点。自然您可能因为某些原因不想将 Web 站点连接到数据库上。本节将讨论创建基于数据源的 Web 站点的优缺点。

1.1.1　优点

创建数据驱动的 Web 站点有许多间接好处，例如可以重用其他项目中的功能，可以跨系统共享通用信息——这些在创建第二个或第三个 Web 应用程序时就可以实现。这里将介绍一些创建数据驱动的 Web 站点的直接好处。

- 内容的质量和及时性——创建数据驱动的站点，最直接的优点就是能及时将新信息显示

在 Web 上，能够放入控件中来保证该信息的质量。每次价格改变或者添加新产品时，我们不必用 Web 设计器创建一个包含该信息的页面后将它再次上载，而是创建一个工具，通过简单地修改数据库来及时发布新的或更新的信息。这是 Web 比传统媒体优越的一个主要地方——可以实时查看信息，而不是原有的数据。通过强制一些规则，如谁可以添加和修改数据，如何进行检查以及是否被认可，可以在发布之前以更严格的方式对数据进行验证，以确保用户只看到正确的信息。

- 功能——将站点需要的所有数据存储在数据库中的另一个主要优点就是改进的功能，这是就用户可以在系统上执行的动作而言。它不是生成“目录”，像这本书那样用索引和目录表作为搜索的手段，而是创建窗体以便允许用户指定要查找的内容，让系统从数据库查询该信息。搜索引擎就是个很好的例子。没有数据库，这类站点只能提供其他 Web 站点的人工目录，需要用户在庞大的页面结构中导航。
- 维护——将站点的数据与表示代码分开存储，那么就不再需要在站点的相关部分之间维护 HTML 文件中的静态链接，也不需要每次重新设计站点时重新应用许多页面的格式和菜单结构。在数据驱动的系统中，Web 页一般是全部页面类的模板，而不是一个信息一个页面。

作为一个例子，可以假定屏幕上显示的页面显示了待售产品的信息。在数据驱动的系统中，这并不是一个单独的 HTML 页面，它是一个包含字段和表的页面，而字段和表可以用于任何产品相关的数据填充。这意味着每次重新设计时只要做很少的工作。类似的，各种不同信息之间的关系也可以存储在数据库中(而不是在页面中硬编码)，可以迅速实现到相关产品和其他信息的链接。

1.1.2 缺点

尽管创建数据驱动的 Web 站点有很多优点，但是其中的一些代价很昂贵，而且数据驱动的 Web 站点并不总是解决问题的最好办法。为了给终端用户提供更丰富的操作，还有一些缺点必须克服，在创建之前考虑这些缺点还是很重要的。

- 开发——大部分数据驱动的 Web 站点一开始都是静态的，而且如今仍然有许多静态站点要创建。想显示的内容并不适合于数据驱动的站点，创建数据驱动的系统要求额外的时间和技巧，这些都使得开发的产品更复杂，更容易出错(不可避免的)。我们不得不把这些代价与这类系统的优点作一下衡量。
- 性能——数据驱动的 Web 站点的性能很容易出现问题。如果站点完全是静态的，那么在组织系统时或是怎样可以满足更多用户的需要时没有任何限制。提高性能最简单的方法就是购买更快的处理器和更多的内存。当这些条件满足时，就可以创建多种站点，用户就可以被重定向到负载最小的站点。这可以以线性形式继续下去，每次添加新的 Web 服务器都会对应提高性能。

有了数据驱动的站点，情况就不是这样，因为整个系统都依赖于一个资源：数据库。如果没有谨慎设计的话，那么数据库会造成系统中的“瓶颈”，因此在等待检索系统时会使得应用程序停滞不前。而要解决这个瓶颈问题是个很困难的工作——用几个同步的数据库是解决这一问题的一个方法，但这代价是很昂贵的，而同步过程中的系统开销也是很大的。

- 代价——除了上面提到的技术问题，还有一些相关的商业问题。对于相对静态的站点来说，创建数据库和编写访问它的代码所需的时间就比仅仅编辑 HTML 页面要多。而且企业级数据库系统本身很昂贵。仅考虑 Microsoft 的数据存储方案的话，那么众所周知，使用 SQL Server(Microsoft 的企业级数据库服务器)生成方案所提供的优点比 Access 的更多(它的桌面数据库)，例如更高的性能，对产业标准的更好支持，但其代价也更昂贵。

1.2 数据源

读者现在已考虑了上面的一些问题或所有问题，如果想编写数据驱动的 Web 应用程序，首先需要说明的就是最终出现在用户屏幕上的信息的来源。根据数据类型，对数据的操作以及系统使用量这些因素，有多种选择。本节描述了赞同和反对使用 3 种最常用的数据源类型的原因，还对其他类型作了一个概述。

1.2.1 数据库

在开始考虑数据源时，最先想到的是数据库，它通常都提供了最可靠、伸缩性最大和最安全的数据存储。在处理大量数据时，数据库也提供了最好的性能。然而也有事实表明在有些情况下，它们并不是最好的选择。

一般来说，数据库存储大量数据的方式就是以任意的顺序检索任意数量的数据。对于小的数据集合来说，例如一组联系方式，那么创建和访问数据库时所用的时间及其他花费都可能会抹杀数据库所提供的优点。

下一章将更多地讨论数据库的结构，但作为一个很简单的例子，将公司雇员的一些信息存储在数据库中需要创建 Employee 表，这个表可以包含许多雇员的同组数据。这些信息可以包括他们的 EmployeeID(号码)、LastName、FirstName、BirthDate 和 Country。见图 1-1。

图　1-1

注意：

本章将通过比较和演示来说明如何存储和显示数据。为了前后连贯，将使用同一个例子：那就是存储某公司雇员的资料。

在显示数据库的图表，与其他数据源的图表作比较时，要注意的一点就是它是基于一个存储信息的模型，而不是数据的例子。而数据库实际存储信息的方式基本上是隐藏起来的，但是我们是描述概念而不是实际的数据项。

1.2.2 文本文件

与用数据库为 Web 站点存储数据相对的节省空间的方式是使用文本文件。尽管文本文件可以用想得到的几乎所有的格式存储信息，但它们一般用于存储一组数据，每一行一个数据项。如果想获取上面的雇员信息，那么可以按如下所示的方式，将两个雇员的 LastName、FirstName、BirthDate 和 Country 存储在文本文件中：

```
Smith, John, 05-04-1979, UK
Bloggs, Joe, 29-09-1981, US
```

对于这样简单的信息，文本文件提供了读取和写入数据的简单方式。如果存储的数据有更多的结构，那么就要花费更多的时间。例如，每个雇员都订购一些办公用品。那么就不是向文本文件中添加所有这些信息，更好的办法就是分别保存，接着定义这两组数据之间的关系。

如果数据的“结构”是这样的，那么必须有给该文件一些结构的方法，必须实现在内存中检索和表示它的方法。其中的一种方法就是使用 XML。

1.2.3 XML

在某些方面来看，XML 文档可以看作文本文件和数据库之间的过渡；它们使用文本文件存储数据，但使用分层的、关联的格式，它们是可扩展的和自描述的，提供了数据库系统的许多优点。在进一步讲述 XML 用作数据源之前，还是看一下 XML 文档的示例代码段：

```
<company>
  <employees>
    <employee LastName="Smith" FirstName="John"
              BirthDate="05-04-1979" Country="UK" />
    <employee LastName="Bloggs" FirstName="Joe"
              BirthDate ="29-09-1981" Country="US" />
  </employees>
</company>
```

正如所见，存储示例信息的方式与在文本文件中存储一样，但还有对该信息的本质的说明。其中，29-09-1981 是 Joe Bloggs 的 BirthDate，因为数据就是这样说明的。XML 的另一个优点就是它可以在一个文档中包含多种信息；可以在<employees>的后面插入下列代码：

```
<orders>
  <order ID="1">
    <product>Staples</product>
    <product>Pencils</product>
  </order>
  <order ID="2">
    <product>Biros</product>
    <product>Erasers</product>
  <order>
</orders>
```