Broadview
www.broadview.com.cn

国际技术期刊中文精华合集

Visual Studio Magazine

2002-2003 中文精华合集 [第1辑]

Fawcette Technical Publications 独家授权

国际技术期刊合集编委会 编译

- 国际名刊精华
- 主流技术荟萃
- 专业开发必读

電子工業出版社
PUBLISHING HOUSE OF ELECTRONIC INDUSTRY
http://www.phei.com.cn

国际技术期刊中文精华合集

Visual Studio Magazine

2002—2003 中文精华合集 [第1辑]

Fawcette Technical Publications 独家授权

国际技术期刊合集编委会 编译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书汇集了国际著名技术媒体 Fawcette Technical Publications 旗下技术期刊 Visual Studio Magazine 2002—2003 年度精华文章数十篇。主要围绕微软公司的 Visual Studio 开发套件，涉及了 Visual Basic、Visual C++、Visual Basic.NET、Visual C#.NET 等当前主流的开发工具，由各自领域内的一流专家撰写，其内容包括了从编程技术到项目组织管理，从工具使用技巧到新技术剖析的各个方面，技术含量丰富，观点权威，涵盖面广。

本书不仅适合专业软件开发者阅读学习和参考，同时也适合广大技术爱好者、在校学生和教师阅读学习。

版权贸易合同登记号　图字：01-2003-7709

图书在版编目（CIP）数据

Visual Studio Magazine2002—2003 中文精华合集．第 1 辑 / 美国发赛特技术出版集团著；国际技术期刊合集编委会编译．—北京：电子工业出版社，2004.4

（国际技术期刊中文精华合集）

书名原文：Visual Studio Magazine

ISBN 7-5053-9717-6

Ⅰ.V...　Ⅱ.①美...②国...　Ⅲ.计算机网络—程序设计　Ⅳ.TP393

中国版本图书馆 CIP 数据核字（2004）第 015381 号

责任编辑：孟　岩

印　　刷：北京中科印刷有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱　邮编　100036

经　　销：各地新华书店

开　　本：880×1230　1/16　印张：22.25　字数：744 千字

印　　次：2004 年 4 月第 1 次印刷

印　　数：8000 册　　定价：32.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

前　言

这是一本开发技术文集，收录了国际著名开发技术期刊 Visual Studio Magazine 杂志 2002—2003 年发表的一批精华文章。

程序员作为一个职业群体，不但需要系统学习基础和经典理论，在实际项目中不断磨练自己，而且需要与同行充分交流，保持对新技术和新思想的敏感，向同行中的佼佼者取经。在这一点上，技术期刊能够起到别的载体无法替代的作用，而国外的高水平技术期刊，在这方面更是有着辉煌的传统。技术期刊中的文章，一方面能紧跟最新的技术趋势，另一方面来自名家之作，高屋建瓴，能够反映比较高的技术水平和崭新的视角。在 IT 产业发展的历史上，技术期刊起到了重要的推动作用。很多重要的新技术和新思想，最初都是发表在期刊上，进而对整个产业界产生深刻影响。因此，作为一名不断追求进步的程序员，应当经常地阅读高水平的国际技术期刊。

Fawcette Technical Publications（发赛特出版集团，简称 FTP 集团）是美国著名的 IT 技术媒体和出版集团，旗下拥有著名的 VSLive!技术大会以及 Visual Studio Magazine、Java Pro Magazine、.NET Magazine 和 XML Web Services Magazine 等技术期刊。FTP 集团的技术期刊拥有一大批世界知名的技术作家，如 VB 之父 Alan Cooper、.NET 专家 Dino Esposito、Java 专家 James Cooper 等，一向以文章质量高、观点权威新颖、技术含量丰富著称。

微软公司出品的 Visual Studio 开发套件，是当前专业软件开发工具市场上最主流的开发工具，其中的 Visual Basic、Visual C++、Visual Basic.NET 和 Visual C#等开发工具均成为专业软件开发市场上炙手可热的产品。

Visual Studio Magazine 的前身是非常成功的 Visual Basic Programmer's Journal 和 Visual C++ Programmer's Journal。为了适应技术变革，这两本著名的技术期刊合并成 Visual Studio Magazine。合并后内容更加集中，技术水平更高更权威。此次我们本着为广大专业开发人员和学习者服务的宗旨，引进 Visual Studio Magazine 杂志，将其精华内容集结成书，翻译出版。在选编时，我们尽可能注意技术期刊内容的特点，既选择一些最新、最前沿的技术文章，也选择了一些立意深远、具有长期意义的文章。希望这本精华合集既能够满足开发者平时学习工作的需要，也能够具有一定的保存价值。这是我们的一个尝试，希望得到广大读者的积极反馈。

全书共 13 篇，分别收录了 Visual Studio Magazine 2002 年 3 月—2003 年 2 月的各期内容，以及 2003 年 3—9 月的文章精选。

本书由“国际技术期刊合集编委会”主持编译，由曾毅、伍炼、彭峥、于景晨等进行选编和翻译。

国际技术期刊的选、编、译，对于我们来说是一个崭新的尝试，限于经验和水平，失误之处在所难免，恳请读者不吝赐教。

国际技术期刊合集编委会

2004 年 2 月

译 者 序

自从微软公司推出了 Visual Studio 的第一个版本后，这个集成软件开发工具就成为很多程序设计人员的最爱。从第一个版本一直到 Visual Studio 6.0，每一次的升级都令开发者们为之心动。而 Visual Studio .NET 更是微软.NET 战略的旗舰，主要用于构建和部署功能强大且安全的 Microsoft .NET 软件。它的推出标志着微软开发工具的革命。

国内很多读者都希望能看到国外专业技术领域的期刊，而翻译出版技术杂志的中文版在国内一直就没有为出版界所重视。为了使读者们能够接触到较新的技术文献与鲜活的范例，博文视点公司特别引进了国外著名技术期刊《Visual Studio Magazine》以满足读者的需求。这本杂志是 FTP 集团的旗帜产品，其前身是著名的《Visual Basic Programmer's Journal》。杂志中涵盖了 VB6、VB.NET、C#和 VC++.NET 等丰富的内容，不乏一些优秀的作品。

为了做好《Visual Studio Magazine》中文合集的编译工作，我们特别组建了翻译技术团队，并由我来主要负责全书的审校与统稿。整个翻译过程十分辛苦，从文章选择到文章翻译、审校，一直到最后的定稿、出版，无不体现所有人员的共同努力。参与翻译的人员有：伍炼、彭峥、于景晨、姚军、周翔、付仲恺、吕阳、李琰、周小卫、雷均、刘欢、王茜、于懋和我。

作为一个期刊杂志的合集，我们不求能使读者在阅读本书之后对整个开发平台有系统、深入的了解。我们只希望通过我们的努力，读者能从本书中获得一些非常实用的 Visual Studio 以及 Visual Studio.NET 开发平台的开发方法与思想。文章涉及的源代码可以通过访问 FTP 集团的主页（http://www.ftponline.com/vsm/）获得。对于程序设计类技术文章来说，专业术语的翻译一直就有不少争议，我们尽力去避免这些问题，本书中的一些专业术语的翻译参考了微软公司官方推出的最新术语对照表。为了便于读者阅读，我们在文章的末尾提供了一些专业术语的中英文对照表。由于译者水平有限，书中难免存在一些问题，不妥之处敬请读者们提出宝贵意见。在使用本书时如果遇到了什么问题，可与杂志社或者通过下面的网址与我们取得联系：http://www.cstc.net.cn。衷心的祝愿每一位读者在 Visual Studio 程序设计之路上旅途愉快。

在中文合集的翻译期间，我和伍炼先生很荣幸地被授予 2003 年微软最有价值专家（Microsoft MVP）称号。作为 Visual Studio.NET 的 MVP，我们更深感一定要为读者做好这本国外技术期刊的中文合集，以回报大家对我们的信任。文章翻译期间，博文视点和《程序员》杂志社编辑部的众多工作人员为本书的出版做了大量的工作，在此一并表示感谢。

曾毅

于 计算机科学技术网

目 录

Visual Studio Magazine

第一篇

2002年3月

- 招聘真正的.NET天才
- 突破SQL的速度障碍
- 扩展关于对话框（About Box）
- 为Non-COM程序添加对象模型

招聘真正的.NET 天才

● Josh Lane

我们大部分人对混乱或悲剧性的事物都有着奇怪的看法。一些人不断地看 CNN，又有些人对车祸有病态的迷恋。而我也不例外，我常常观察那些努力聘请和留住熟练软件开发人员的技术公司。

我们要承认：很多公司在鉴别和雇用编程人才方面做得很差。当然，有一些公司不是真的想聘请最好的员工，但还是有很多公司想努力做到，却仍以失败收场的。具体原因有很多，包括缺乏上层管理的认真对待、缺乏时间组织面试、对面试的准备不足，以及过度相信猎头公司等。但是还有一个更普遍的问题导致这些公司招聘失败，就是经理雇请他们，是基于他们非实质的能力，而开除他们，却因为他们缺乏必要的（或相关的）技能。

最近我重读费瑞德·布鲁克斯教授的一篇经典评论时，想起了这个问题。《No Silver Bullet—Essence and Accidents of Software Engineering》里面说到现在软件开发问题有两种："次要的"和"本质的"。次要问题是有限和短暂的，在数年内就很容易变得不切题，就像在 Windows 2000 里用 C++进行异步输入/输出一样。而本质问题更加深奥，永远相应，也就是说和具体细节问题无关。这些问题包括资源的理想分配、数据的有效传递和精简的系统设计等。

即将来临的".NET 波"使我又一次想起"次要的"和"本质的"问题。我们将会被.NET 工作需求、伴随的.NET 会议和.NET 书籍风暴所淹没。到 2002 年的夏天，我估计将会有公司要聘请"有三年 Web 服务经验"或"三年 C#开发经验"的人。我也不知道到时该笑还是该哭。

但不要误会我，我是很爱.NET 的。它的广度和深度包含了很多巧妙的东西。不过如果揭开销售的面纱，就会发现.NET 只是一套工具，强大的更完善的工具，但还只是工具而已。你不会因为一个人会使用锤子就雇用他帮你修房子，招聘开发者也是同样的道理：如果你想花钱雇用到最好的.NET 开发者，就应该以他们的实质能力做依据。

不要问应聘者如何在.NET 里面连接字符串，而应该要求他们描述 StringBuilder 的优点；不要问他们是否构建过 Web 服务，而应该要他们讲出哪些问题 Web 服务是不能解决的，并解释为什么；问一下.NET 代理和 C++功能指针的区别，要求他们描述 single-writer/multiple-reader 的语义，然后问有关.NET 标准执行的问题。能够从经典的 performance/resource tradeoffs 学到知识，使用简单而非取巧的设计，还有天生就具备的知识抽象能力，这就是开发者具有使用.NET 能力的特征，而不用管他们是否有"3 年 Web 服务经验"。

软件系统的本质不是 C#、Java、MSMQ、COM+或者 SQL Server，相反它是一套相互交织的概念，这些概念授予了系统功能、性质和值。这些都是专业的概念，它们不容易量化、操作、论述、出售、测试、管理或从中建立出某事物。极少软件专业人能够轻松掌握这些概念，就更不用说非软件专业人了，但世界上伟大软件建立起来的原料恰恰是这些概念。

布鲁克斯教授认为对这些概念的掌握程度区分了真正的天才软件开发者和那些业余爱好者。这点我再认同不过了。因此，如果你想为近期内要开发的工程增加.NET 人才，就要听从布鲁克斯教授的建议：问应聘者基于实质性技能的问题。这些问题很难提出来，因为你自己要先有正确的答案；而你的回报是顶尖人才的流入和.NET 开发的美好前景。

突破 SQL 的速度障碍

用 SqlClient 直接访问 SQL Server 数据。

● Dan Fox

开发工具：VB.NET，SQL Server 2000，ADO.NET

ADO.NET 是非常灵活的，对于直接存取 SQL Server 数据提供了 ODBC 甚至是 OLE DB 的快速选择。ODBC 和 OLE DB 在不能直接到达数据源时可以很好地工作，但是通过使用 SQL Server 和 Visual Studio.NET 再配合上 SqlClient.NET Data Provider 可以达到更好的效果。最终的结果是加速应用程序的开发速度和产生的 SQL 应用程序的速度。

在 ADO.NET 中通过托管提供程序.NET 数据提供程序访问数据源，托管类的集与 back-end 数据存储区通信。这些类跟随一个以类和接口为基础的共有设计模式，它们主要存放在 System.Data 和 System.Data. Common 命名空间下。VS.NET beta 2 中提供了两个托管提供程序，通过 OLE DB 提供程序进行访问的 OleDb System.Data.OleDb 和对 SQL 服务器访问的 SqlClient System.Data.SqlClient。从 MSDN 可以下载到 ODBC 托管提供程序的 beta 版。

（参考资料：
Dan Fox《Pure Visual Basic》（第 14 章）[Sams, 1999, ISBN: 067231598X]
《Building Distributed Applications with Visual Basic .NET》（第 7 章）[Sams, 2002, ISBN: 0672321300]
http://msdn.microsoft.com/library/?url=/library/en-us/dndotnet/html/adonetprogmsdn.asp
http://msdn.microsoft.com/library/?url=/library/en-us/dndive/html/data03222001.asp
http://msdn.microsoft.com/vstudio/nextgen/technology/adoplusdefault.asp
http://msdn.microsoft.com/downloads/sample.asp?url=/msdn-files/027/001/668/msdncompositedoc.xml）

OLE DB 和 ODBC 提供连接多种数据源的强大功能。相反，仅仅使用 SqlClient 托管提供程序连接 SQL 服务器 7.0/2000。我预测第三方供应商将为其他特殊的数据存储区建立严密的提供程序。通过这些托管提供程序的类编程，将能够提供最好的性能和公开数据存储区特性，就像 SqlClient 托管提供程序专为 SQL Server 工作一样。

关于 GUIDs

ADO.NET 让使用 GUIDs 代替“系统设定”键的 IDENTITY 更简单。

熟悉 ADO 2.x 的开发者会觉得 SqlClient 公开的对象模型非常舒服（如图 1 所示）。SqlClient 负责 SqlConnection、SqlCommand 和 SqlParameter 对象的实现，类似于 ADO 的 Connection、Command 和 Parameter 对象。当 SQL Server 出现警告或者错误提示时，SqlClient 会填充一个 SqlError 对象集合，该集合包含在 SqlException 对象中，可以使用 Common Language Runtime'sCLR's 结构化异常操作 SEH 轻松地处理。最后，调用 SqlConnection 类的 BeginTransaction 方法，将事务 transaction 实现成独立的类 SqlTransaction。

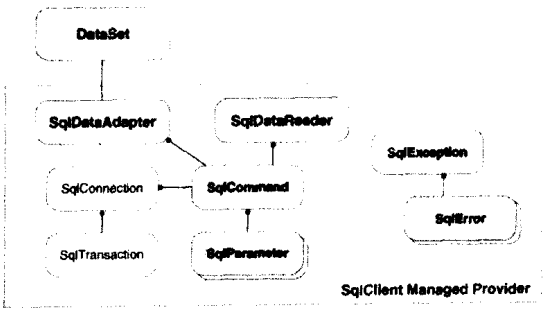

图 1 SqlClient 对象模型

从图 1 中可以看到 SqlClient 命名空间的主要对象以及它们的关系。注意 DataSet 对象不被托管提供程序实现，SqlDataAdapter 会填充它。

和 ADO 一样，SqlClient 类也支持事件。例如，SqlConnection 类的 InfoMessage 事件允许捕获信息消息，像从 SQL 服务器返回的 Print 语句 statement。类似地，SqlDataAdapter 包括的事件有 RowUpdating，它会在更新数据源时触发。

严谨的托管提供程序（如 SqlClient）在管理 back-end 数据存储区通信的所有方面都非常有效率。SqlClient 不使用基础的 OLE DB 提供程序，它通过 SQL 服务器的 Tabular Data StreamTDS 协议直接和服务器通信。在 beta2 的测试中，SqlClient 的运行速度比在 VC++代码中使用 SQL 服务器 OLE DB 提供程序快 10%，也比在 VB 6 中使用 ADO 快 10%。

那么这些效率的价值何在呢？SqlClient 不能使用 OLE DB 会话池 session pooling 汇集数据库连接，它必须使用 COM+pooled 组件特性来运作自身的连接池。使用连接字符串属性可以配置池，如表 1 所示。

表 1　通过属性控制连接池

属　性	默认值	说　明
Connection Lifetime	0	一个连接在池中的生命周期（以秒为单位）
Connection Reset	True	若设为 **False**，则一次往返不会使连接复位，但数据库的上下文可能已经变化
Enlist	True	若设为 **True**，则在存在的分布式事务中应用，若设为 **False**，可提高性能
Min, Max Pool Size	0, 100	池的大小
Pooling	True	可从池中取出连接

通过 SqlConnection 打开 SQL 服务器时，可以将这些属性放置在连接字符串中。注意现在可以指定放置在池中的最小连接数目，因为 SqlClient 使用 Component 服务。

SqlClient 通过实现 SqlDataReader 和 SqlDataAdapter 类也可以提高速度（如图 1 所示）。SqlDataReader 允许以只进和只读方式循环访问从一个服务器返回的集的一行。这项工作很像 ADO 允许使用 fire-hose 游标 cursor 或者 cursorless 结果集。从服务器返回的行都以流的形式进入客户端网络缓冲以便继续检索，这样就以占用客户端和服务器最少资源的方式提供了最快的数据访问速度。

SqlDataAdapter 是 DataSet 对象和 SQL 服务器通信的中间层。这个类存储并使 SqlCommands 被用来在 DataSet 设置的 SQL 服务器中进行选择、插入、更新及删除数据操作。该托管提供程序体系把 ADO Recordset 对象分成两个主要功能，一个是担当数据（实例，非连接的记录集）中间层缓存，另一个是循环访问结果集。可以分别使用 DataSet、SqlDataAdapter 和 SqlDataReader 类的组合。

技巧：使用存储过程

在 SQL 服务器中使用存储过程始终是一个好的操作数据编程习惯。

这里将展示如何使用 SqlClient 与 SQL 服务器工作，借助存储过程读取和写入以 SQL Server 2000 为基础的 Northwind 例子数据库中的产品信息数据。

SqlClient 允许通过 SqlCommand 对象执行存储过程，它的 CommandType 属性还能执行动态 SQL。比如，假设有一个名为 NwindProducts 的数据访问类，从 ASP.NET 应用程序的用户接口可以调用它（如程序 1 所示）。该类包含一个接受连接字符串的构造函数，连接字符串存放在 _connect 私有变量中。将 Option Strict 和 Explicit 设置为 "on"，使用 Imports 语句导入 System.Data.SqlClient 命名空间。VS.NET beta2 将默认地向 System.Data 程序集添加引用。

VB.NET · 用 SqlCommand 执行存储过程

SqlClient 允许通过 SqlCommand 对象执行存储过程，它的 CommandType 属性还能帮助执行动态 SQL。在这里，拟通过 ASP.NET 应用程序用户接口调用数据访问类 NwindProducts。注意这里是如何实现 AddCategory 方法的。

程序 1.　在这里，通过 ASP.NET 应用程序用户接口调用数据访问类 NwindProducts。注意这里是如何实现 AddCategory 方法的。

```
Option Strict On
Option Explicit On

Imports System.Data.SqlClient
Imports System.Data.SqlTypes
Imports System.IO
Imports System.Xml
Public Class NwindProducts
```

```
Private _connect As String
Public Sub New(ByVal connect As String)
  _connect = connect
End Sub
Public Function AddCategory( _
  ByVal name As String) As Integer
  ' Verify the incoming parameters
  If name Is Nothing Then
    Throw New ArgumentNullException( _
      "name ", "This method requires name")
  End If
  Return _AddCategory(name, Nothing)
End Function
Public Function AddCategory(ByVal name As String,_
  ByVal description As String) As Integer
  If name Is Nothing _
    Or description Is Nothing Then
    Throw New ArgumentNullException( _
      "name or description", _
      "Name and description are required")
  End If
  Return _AddCategory(name, description)
End Function
Private Function _AddCategory( _
  ByVal name As String, _
  ByVal description As String) As Integer
  ' private implementation
  Dim cn As New SqlConnection(_connect)
  Dim cm As SqlCommand
  Try
    ' Build the command
    cm = New SqlCommand("usp_AddCategory", _
      cn)cm.CommandType = _
      CommandType.StoredProcedure
    ' Add the parameters
    Dim parm As New SqlParameter("@name", _
      SqlDbType.NVarChar)
    parm.Value = name
    cm.Parameters.Add(parm)
    If not description Is Nothing Then
      Dim parm2 As New _
        SqlParameter ("@desc", SqlDbType.NText)
      parm2.Value = description
      cm.Parameters.Add(parm2)
    End If
    Dim retparm As New _
      SqlParameter("RETURN", SqlDbType.Int)
    retparm.Direction = _
      ParameterDirection.ReturnValue
    cm.Parameters.Add(retparm)
    ' Execute the command
    cn.Open()
    cm.ExecuteNonQuery()
    Return CType(retparm.Value, Integer)
  Catch e As Exception
    Throw e
  Finally
    ' Release connection
    cn.Close()
  End Try
 End Function
End Class
```

NwindProducts 中为 AddCategory 方法重载了两个标记，name 参数只接受要添加到数据库中的新种类名称，另一个是添加描述。这两个标记实际上应用于_AddCategory 以完成数据库工作，当参数没有被添加时 ArgumentNullException 异常将被抛出。

注意： AddCategory 中存储过程的调用，如程序 1 所示。SqlCommand 对象被实例化并且传递存储过程名以调用，连同一个 SqlConnection 对象的引用。在设定 CommandType 属性后，SqlParameterCollection 被添加到 SqlParameter 对象中。SqlParameter 类被重载的构造函数用在这里接受参数名和它的类型。本例中，其他的属性都包含默认值，使用 Value 属性设定值时都将被使用。在 SqlDbType 枚举类型中可定义数据类型，并且可直接映射到 SQL 服务器数据类型中。使用 Add 方法可以将参数增加到 Parameters 集合中。

通常如果要捕获存储过程的返回值或者使用 OUTPUT Transact-SQLT-SQL 关键字定义的参数，可以简单地定义一个 SqlParameter 对象，把 Direction 属性分别设置为：

ParameterDirection.ReturnValue 或者

ParameterDirection.Output。

在_AddCategory 方法中，存储过程的返回值会包含由 Categories 表中 IDENTITY 列分配的新的 CategoryID。

将参数增加到 SqlParametersCollection 对象中的时候，不需要以任何特别的顺序，甚至返回声明也如此，因为 SqlCommand 对象执行存储过程是通过参数名而不是参数位置来使用参数。当然，也不一定增加参数到集合中，默认情况下为空。

例如，当 EXEC 语句不提供时，usp_AddCategory 存储过程默认@desc 参数为空。

```
CREATE PROC usp_AddCategory
@name nvarchar(15),
@desc ntext = NULL
AS
INSERT INTO Categories
(CategoryName, Description)
VALUES (@name, @desc)
RETURN @@identity
GO
```

最后，调用 SqlConnection 对象的 Open 方法与 SQL 服务器建立连接，调用 ExecuteNonQuery 方法执行存储过程，将有效的行数返回服务器中。使用 ExecuteNonQuery 可以在存储过程没有返回结果集时保存客户端的资源。由于存储过程返回新的 CategoryID，因此可以通过访问从该方

法返回的 SqlParameter 对象的 Value 属性读取它。所有放在 Try Catch 块中的 SQL 服务器访问代码，可以方便地将异常返回给调用者，并在 Finally 语句中关闭连接。在客户端需要类似如下的代码来调用 AddCategory 方法：

```
Dim prod As New NWindProducts( _
  "server=ssosa;database=northwind;uid=sa")
Dim catID As Integer
catID = prod.AddCategory("Waffles", _
  "A tasty breakfast")
```

通过建立一个嵌入式 EXEC 语句的字符串，以及设置 CommandType 属性为 CommandType.Text 来明确地使用参数，以执行一个存储过程。这样将使得 SqlClient 通过远程过程调用 Remote Procedure Call 请求调用存储过程。由于它们回避了语句分析和参数处理，直接处理 EXEC 语句，所以 RPC 请求运行得更快。尽管在存储过程中不经常使用，但 SqlCommand 对象仍然支持 Prepare 方法。可以调用它在服务器上预备动态 SQL，以提高后面相同语句的执行速度。sp_prepexec、sp_execute 和 sp_unprepare 系统存储过程都被后台调用。

除了使用 ExecuteNonQuery 方法，SqlCommand 对象还支持 ExecuteScalar 方法。这个方法不保存服务器的处理时间，但是它提供了简便的方法捕获单独的一行或者一列结果。例如，可以使用在存储过程中的聚合函数 usp_GetCustTotal 返回一个指定客户的销售总额。

```
CREATE PROC usp_GetCustTotal
@customerID nchar(5)
AS
SELECT SUM(UnitPrice * Quantity * (1-Discount))
FROM [Order Details] a JOIN Orders b
ON a.OrderID = b.OrderID
WHERE b.CustomerID = @customerID
GO
```

使用 ExecuteScalar 方法将值作为对象返回，优于缓存某个 DataSet 中单独的一行或者列的结果集或者打开一个 SqlDataReader。可以将它们转变成适当的类型（如程序 2 所示）。

VB.NET · 通过 ExecuteScalar 进行说明

程序 2. ExecuteScalar 方法使捕获单独的行或列变得简单。ExecuteScalar 方法将值作为对象返回，并能转换为适当的数据类型。在本例中使用 Decimal 数据类型返回当前的值。

```
    Public Function CustomerTotal(ByVal customer As String)
As Decimal
      Dim cn As New SqlConnection(_connect)
      Dim cm As SqlCommand
      If customer Is Nothing Then
        Throw New ArgumentNullException("customer", _
          "Customer must contain ID")
      End If
      Try
        ' Build the command
        cm = New SqlCommand("usp_GetCustTotal", cn)
        cm.CommandType = CommandType.StoredProcedure
        ' Add the parameters
        Dim parm As New SqlParameter("@customerID",
SqlDbType.NChar)
        parm.Value = customer
        cm.Parameters.Add(parm)
        cn.Open()
        ' Rounding because it's a monetary value
        Return Decimal.Round(CType(cm.ExecuteScalar(),
Decimal), 2)
      Catch e As Exception
        Throw e
      Finally
        cn.Close()
      End Try
    End Function
```

读取和写入

除了 ExecuteScalar 方法，还可以借助 SqlDataReader 方法或者填充一个数据集的方法读取 SQL 服务器中的数据。可使用 SqlCommand 对象的 ExecuteReader 方法建立 SqlDataReader 游标。例如，在 .NwindProducts 类中用 ListCategories 方法调用存储过程 usp_GetCategories，这将向数据库发出一个简单的 Select 语句，代码如下：

```
  Public Function ListCategories() As SqlDataReader
    Dim cn As New SqlConnection(_connect)
    Dim cm As SqlCommand
    ' Build the command
    cm = New SqlCommand("usp_GetCategories", cn)
    cm.CommandType = CommandType.StoredProcedure
    cn.Open()
    Return
cm.ExecuteReader(CommandBehavior.CloseConnection)
  End Function
```

不需要插入 Try Catch 块以使 Finally 语句能够关闭连接。ExecuteReader 从 CommandBehavior 枚举中接受一个位组合，当 SqlDataReader 关闭时关闭连接。这样就能够在各方法间传递 SqlDataReader，或从一个未整理的基础连接中返回它。无论哪种方法，都不能在应用程序间传递

SqlDataReader，因为SqlDataReader不是来源于MarshalByValueComponent，并能够保持与数据库服务器的连接。当对象在Component Services的服务器软件包中被承载时，不能执行ListCategories方法。

在客户端执行了ListCategories方法后，可以访问被返回的SqlDataReader对象，代码如下：

```
Dim dr As SqlDataReader
Dim catID As System.Data.SqlTypes.SqlInt32
Dim categoryName As _
  System.Data.SqlTypes.SqlString
dr = prod.ListCategories()
Do While dr.Read()
  catID = dr.GetSqlInt32(0)
  categoryName = dr.GetSqlString(1)
  Console.WriteLine(catID.ToString() & _
    " : " & categoryName.ToString())
Loop
dr.Close()
```

这里的SqlDataReader公开的方法为GetSqlInt32，可以通过结果集中的序列号检索每列。该方法检索System.Data.SqlTypes命名空间中的数据类型。这些类型直接映射自SQL服务器数据类型，它提供了更快速、更准确的数据访问。SqlDataReader支持将GetValues和GetSqlValues方法取得当前行的所有列放入一个数组中，但是它不支持任何类似ADO 2.x的GetRows方法，全部的结果集都将被复制到一个多位数组中。

除了手工循环结果集，还可以使用SqlDataReader对象的DataSource属性把结果绑定到表格中或者其他的控件中。事实上，如果存储过程返回多个结果集比如返回静态查找数据，可以使用NextResult方法循环访问每个结果集，并绑定到拖放控件或者其他用户界面元件上。

SqlClient也能够通过ExecuteXmlReader方法填充System.Xml.XmlReader对象。如果存储过程使用FOR XML语句建立一个来自SELECT语句的XML文档，该工作就会启动（如程序3所示）。

SQL · 通过ExecuteXmlReader填充XML对象

程序3. 当GetCatTotals方法使用ExecuteXmlReader检索XML文档时，usp_GetCatTotals存储过程会通过FOR XML EXPLICIT语句建立XML文档。该方法返回打开的XmlReader，可以循环访问并使用其中的方法和属性，如Read、ReadInnerXml和Value。

```
CREATE PROCEDURE usp_GetCatTotals
@CatID int
AS
SELECT 1 As Tag, NULL as Parent,
  @CatID AS [OrderTot!1!CatID],
  NULL AS [Product!2!ProductID],
  NULL AS [Product!2!ProductName],
  NULL AS [Product!2!TotalUnits!element]
UNION
SELECT 2, 1,
  NULL,
  d.ProductID,
  d.ProductName,
  SUM(b.UnitPrice)
FROM Orders a JOIN [Order Details] b
  ON a.OrderID = b.OrderID
JOIN Products d ON d.ProductID = b.ProductID
WHERE d.CategoryID = @CatID
GROUP BY d.ProductID, d.ProductName
FOR XML EXPLICIT
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
Public Function GetCatTotals( _
  ByVal catID As Integer) As XmlReader
  Dim cn As New SqlConnection(_connect)
  Dim cm As SqlCommand
  If catID = 0 Then
    Throw New ArgumentNullException("catID", _
      "CatID must be > 0")
  End If
  Try
    ' Build the command
    cm = New SqlCommand("usp_GetCatTotals", cn)
    cm.CommandType = CommandType.StoredProcedure
    ' Add the parameters
    Dim parm As New SqlParameter("@catID", _
      SqlDbType.Int)
    parm.Value = catID
    cm.Parameters.Add(parm)
    cn.Open()
    Return cm.ExecuteXmlReader()
  Catch e As Exception
    Throw e
  Finally
    cn.Close()
  End Try
End Function
```

用SqlDataAdapter更新

SqlClient支持流访问数据和缓存DataSet对象中的数据。当然，SqlDataAdapter的Fill方法可以填充DataSet对象，不过从DataSet中更新数据是很麻烦的。在本例中，SqlDataAdapter担当了SQL Server和DataSet通信的中间装置（如图2所示）。

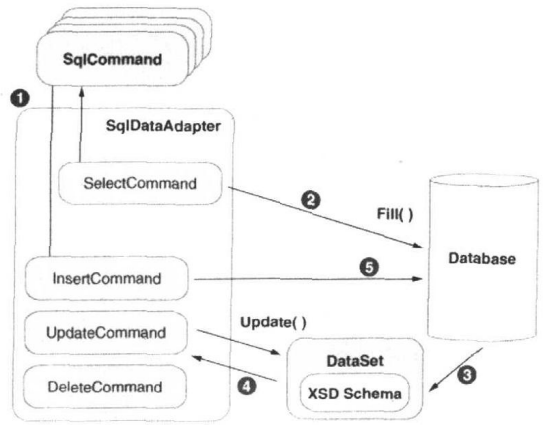

图2 用SqlDataAdapter更改数据

这里可以看到填充和更新数据的处理过程。

(1) SqlCommand对象关联SqlDataAdapter的属性。

(2) 在Fill方法被调用时，命令被执行并且DataSet对象被填充。

(3) 当调用Update方法时，DataSet对象中的数据被映射到Command对象上，并且在服务器上执行修改语句。

更新缓存中的记录数据时必须小心。幸运的是，SqlDataAdapter依靠从SqlCommand对象映射的InsertCommand、UpdateCommand和DeleteCommand属性使这一过程变得轻松。如果用命令填充这些对象将会引用存储过程，当调用Updata方法时SqlDataAdapter也会执行存储过程。例如，SaveSuppliers方法将DataSet作为接受接收，并更新数据库。SaveSuppliers使用两个SqlCommand对象：一个用来插入更新，另一个则用来删除，如程序4所示。

VB.NET · 利用SqlDataAdapter更新缓存中的记录

程序4. SaveSuppliers使用两个SqlCommand对象，一个用来插入更新，另一个则用来删除。

```
Public Function SaveSuppliers( _
  ByVal suppliersData As DataSet) As DataSet
  Dim cn As New SqlConnection(_connect)
  Dim da As New SqlDataAdapter()
  Dim cmSave As New _
    SqlCommand("usp_SaveSupplier", cn)
  Dim cmDel As New _
    SqlCommand("usp_DelSupplier", cn)
  ' Don't do anything if no changes made
  If Not suppliersData.HasChanges Then
    Return Nothing
  End If
  ' Configure the Save command
  With cmSave
    .Parameters.Add(New _
      SqlParameter("@supplierID", SqlDbType.Int))
    .Parameters(0).SourceColumn = "SupplierID"
    .Parameters.Add(New SqlParameter( _
      "@company", SqlDbType.NVarChar))
    .Parameters(1).SourceColumn = "CompanyName"
    .Parameters.Add(New SqlParameter( _
      "@contact", SqlDbType.NVarChar))
    .Parameters(2).SourceColumn = "ContactName"
    .Parameters.Add(New SqlParameter("@city", _
      SqlDbType.NVarChar))
    .Parameters(3).SourceColumn = "City"
    .CommandType = CommandType.StoredProcedure
    Dim retParm As New SqlParameter( _
      "RETURN", SqlDbType.Int)
    retParm.Direction = _
      ParameterDirection.ReturnValue
    retParm.SourceColumn = "SupplierID"
    .Parameters.Add(retParm)
  End With
  ' Configure the Delete command
  cmDel.Parameters.Add(New _
    SqlParameter("@supplierID", SqlDbType.Int))
  cmDel.Parameters(0).SourceColumn = _
    "SupplierID"
  cmDel.CommandType = _
    CommandType.StoredProcedure
  ' Set up the commands
  With da
    .InsertCommand = cmSave
    .UpdateCommand = cmSave
    .DeleteCommand = cmDel
  End With
  Try
    da.Update(suppliersData)
  Catch e As SqlException
    ' An error occurred
    Dim ds As New DataSet()
ds.Merge(suppliersData.Tables(0).GetChanges())
    Return ds
  End Try
End Function
```

使用SqlDataAdapter更新，需要把DataSet对象中相应的列映射到存储过程中。向SqlCommand对象集合增加参数时，SourceColumn属性用于指定DataSet，实际上是DataTable中需要映射到SqlParameter对象中的列。SqlDataAdapter能够从每行取出正确的值并传递到SqlCommand对象内。

更新工作也是直接进行的。retParm对象引用usp_SaveSupplier过程的返回值，它是SupplierID列的映射。SqlCommand被运行时，SqlDataAdapter检索返回值并把它们放到列中，因此在客户端会自动地更新系统分配的IDENTITY。在调用更新方法时，SqlDataAdapter同样建立和关闭SQL服务器连接。

SaveSuppliers方法也提供了错误操作———一键批量更