PROGRAMMER TO PROGRAMMER™

PROFESSIONAL Visual Basic .NET Transactions

Written and tested for final release of .NET v1.0

VB.NET 事务处理高级编程

Matthew Bortniker James Conard 著

袁勤勇 何欣 郑巍 等译

清华大学出版社

http://www.tup.tsinghua.edu.cn

VB.NET 事务处理高级编程

Matthew Bortniker
James Conard
著
袁勤勇　何欣　郑巍　等译

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：01-2002-3020

内 容 简 介

本书涵盖了 ADO.NET，SQL Server，ASP.NET 和 XML 的基本概念；如何创建 Web 服务，并用于事务处理；使用 COM+开发分布式事务处理组件；交互操作性与.NET Remoting；程序调试与故障排除；最后深入剖析了一个案例研究，应用所学的概念和技术构建完整的分布式事务处理系统。本书不仅全面介绍了事务处理的理论，而且还涉及到了与数据访问相关的各种技术。

本书适用于熟悉 VB.NET，并希望在应用程序中融合事务处理技术的程序员。

Matthew Bortniker,James Conard: Professional Visual Basic.NET Transactions
EISBN：1-861005-95-4

图书在版编目(CIP)数据

VB.NET 事务处理高级编程/(美)波特尼科，(美)科纳德著；袁勤勇等译。—北京：清华大学出版社，2002
书名原文：Professional Visual Basic .NET Transaction
ISBN 7-302-05619-6

Ⅰ.V…Ⅱ.①波…②科…③袁… Ⅲ.BASIC 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字 (2002) 第 043719 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.tsinghua.edu.cn
责任编辑：于平
印 刷 者：北京密云胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：30.25　**字数**：774 千字
版　　次：2002 年 7 月第 1 版　2002 年 7 月第 1 次印刷
书　　号：ISBN 7-302-05619-6/TP・3314
印　　数：0001～5000
定　　价：52.00 元

出版者的话

近年来，国内计算机类图书出版业得到了空前的发展，面向初级用户的应用类软件图书铺天盖地，但是真正有深度和内涵的高端图书不多。已经掌握计算机和网络基础知识的人们，尤其是 IT 专业人士迫切需要“阳春白雪”。IT 图书市场呼唤精品！

为了满足这种市场需求，清华大学出版社从世界出版业知名品牌 Wrox 出版公司引进了受到无数 IT 专业人士青睐，被奉为 IT 出版界经典之作的 Professional 系列丛书。这套讲述最新编程技术与开发环境的高级编程丛书，从头到尾都贯穿了 Wrox 出版公司“由程序员为程序员而著(Programmer to Programmer)”的出版理念，每一本书无不是出自软件大师之手。实际上，Wrox 公司的图书作者都是世界顶级 IT 公司(如 Microsoft，IBM，Oracle 以及 HP 等)的资深程序员，他们的作品既深入研究编程机理，传授最新编程技术，又站在程序员的角度，指导程序员拓展编程思路，学习实用开发技巧，从而风靡世界各地，被 IT 专业人士和程序员视为职业生涯中的必读之作。

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并获得相关的技术支持。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。

序　言

0.1　本书读者对象

本书是为希望使用 Visual Basic .NET 进行事务处理的所有人而写。本书将施惠于熟悉 Visual Basic .NET，并且希望在他们的应用中融入事务处理的中级程序员。它也适合于为以前曾使用过 MTS 和 COM+服务，并且希望了解在.NET 的领域中如何实现事务处理的有经验的 Visual Basic 6.0 开发者。

尽管本书会试图涵盖一些已有技术的基本概念，然而，还是希望读者能够比较熟悉 Visual Basic .NET、SQL Server、ASP、HTML 和 XML。如果读者对.NET 的整体状况、或者专门的 Visual Basic .NET 感兴趣，可以求助于清华大学出版社出版的如下书籍：

《.NET Framework》

《VB.NET 高级编程》

0.2　本书涵盖内容

第 1 章，基础知识——事务处理、数据访问以及应用体系结构，介绍了事务处理是什么及其如何工作。该章还讨论了应用的体系结构，并且介绍了访问数据的技术。毕竟，为了处理数据，我们首先必须能够访问数据。

第 2 章，使用 Visual Basic .NET 进行数据访问， 介绍了 ADO.NET，并且讨论了它与 ADO 的区别。该章还介绍了 XML，ADO.NET 可以使用这项技术来确保与任何平台上的任何应用进行有效的数据传输。

第 3 章， SQL Server 和 Visual Basic .NET， 讨论了 SQL Server 以及 SQL 语言。该章还将分析存储过程、怎样在 SQL Server 中进行事务处理，以及怎样建立事务处理型存储过程。

第 4 章， ASP.NET，介绍了使用 ASP.NET 的 Web Forms。由于前面的章节已经涉及了数据层，所以这章将会分析表现(用户)服务。它首先会介绍 ASP.NET 和 Web Forms，然后会讨论 HTML 服务器控件和 Web 服务器控件。这时，将会讨论数据绑定，以及使用 Repeater、DataList 和 DataGrid 控件显示数据。可以使用各种模板对这些数据显示编排格式。

第 5 章， Web 服务、SOAP 和 Visual Basic .NET， 展示了怎样建立和实现 Web 服务，以及怎样将它们用于事务处理。该章还详细分析了 SOAP 的特性，以及实现 Web 服务安全性的技术。

第 6 章，使用 Visual Basic .NET 和 COM+构建分布式应用，讨论了分布式事务处理的概念，以及我们怎样利用 COM+，从 Visual Basic .NET 开发分布式的事务处理型组件。在该章中，我们将会更加详细地深入分析一些第 1 章中讨论的技术。

第 7 章，互操作性， 建立在第 6 章介绍的概念上，将会分析称为 COM 互操作性的功能强

大的.NET Framework 特性，它提供了两种将.NET 组件和应用与基于 COM 的组件和应用结合使用的集成支持。

第 8 章，.NET Remoting 和组件，展示了怎样通过不同的计算机远程访问.NET 程序集，进而使用替代 DCOM 的.NET Remoting 建立完全分布式的.NET 应用。

第 9 章，调试和排除故障，分析了各种调试和排除应用和系统问题的技术。该章首先会讨论用于处理异常的 Try...Catch...Finally 结构的基础知识，然后会讨论 Exception 类的各种属性和方法，它会捕获托管和非托管的代码之间的错误，并且记录错误，还会讨论怎样将异常写入记录(trace)文件，以及怎样测试应用性能。

第 10 章，案例分析：IBuyAdventure.NET 2.0，将我们在前面章节学到的概念和技术应用于一个完整的分布式事务处理系统。这个案例分析的主要目的是展示组织怎样使用.NET Framework 和相关的.NET 技术，建立一个健壮的、可扩展的高性能事务处理应用。除了展示事务处理最实用的内容以外，这个案例分析还会展示各种关键.NET 技术的使用，包括：IIS、COM+、SQL Server 2000、ADO.NET、ASP.NET Web Forms、ASP.NET Web 服务、XML 数据处理，以及 Visual Basic .NET 的新特性，例如结构化异常处理、继承、共享方法和方法重载以及覆盖。

0.3 使用本书所需配置

读者所需的就是让 PC 运行起来：

- Windows 2000
- Visual Studio .NET 专业版(或者更高版本)
- SQL Server 2000
- IIS
- Internet Explorer 5.5 或者更高版本

0.4 客户支持

我们总是希望能够从读者那里获得有价值的信息，希望能够知道读者对本书的看法：您喜欢什么、您不喜欢什么、您希望我们下次能够在哪些地方做得更好。读者可以向我们提出您的意见，您可以使用本书后面的反馈卡，或者也可以通过向 feedback@wrox.com 发送电子邮件。请注意在您的消息中提及本书的名称。

0.4.1 怎样下载本书的示例代码

当用户访问 Wrox 站点 http://www.wrox.com/的时候，只需通过我们的 Search (搜索)功能，或者使用标题列表就可以定位标题。然后单击 Code 栏中的 Download，或者单击书籍明细页面上的 Download Code。

从我们站点下载的文件已经使用 WinZip 进行了归档。当用户将附件保存入硬盘上的文件夹之后，用户需使用 WinZip 或者 PKUnzip 这样的解压缩程序展开文件。当用户展开文件之后，

代码通常会解压缩到章节文件夹中。在用户开始解压缩过程的时候，要确保设置了软件(WinZip、PKUnzip 等)的文件夹名称。

0.4.2 勘误表

我们已经尽了最大的努力来保证在文本或者代码中不出现错误。然而，一切事物不可能完美无缺，一定还会存在错误。如果读者在我们的书中发现了拼写错误或者代码错误之类的问题，我们将非常感激您能够提供反馈。通过发送勘误表，可以节省其他读者遭受挫折的时间，当然，这也能够帮助我们提供更高质量的信息。读者只需将这些信息通过电子邮件发往 support@wrox.com。这些信息将会受到检查，如果正确，会将其张贴在相应标题的勘误页面上，或者用于书籍的后续版本。

为了在 Web 站点上找到勘误表，可以访问 http://www.wrox.com/，然后利用我们的 Advanced Search 或者标题列表定位相应标题。单击位于图书明细页面上的封面图形之下的 Book Errata 链接。

0.4.3 电子邮件支持

如果读者希望直接向详细了解本书的专家咨询本书中的问题，可以将如下信息使用电子邮件发往 support@wrox.com：

- 在主题栏中写入书的名称、ISBN 的最后 4 位数字，以及问题出现的页码。
- 在消息的主体部分，写下您的姓名、联系方法、以及问题。

我们不希望发送给读者垃圾邮件。我们需要详细信息来节省读者和我们的时间。当读者发送电子邮件消息之后，它将会经过如下的支持链：

- 客户支持——读者的信息将会发送给我们的客户支持人员，他们将是最先阅读这些信息的人员。他们拥有常见问题的文档，将会立即回答有关书籍或者 Web 站点的普通问题。
- 编辑——更深层次的疑问会转达给负责那本书的技术编辑。他们对编程语言或者特定的产品富有经验，能够回答相关主题的详细技术问题。
- 作者——最后，不幸的是如果编辑不能够回答读者的问题，他或者她就会将咨询转达给作者。我们会试图避免分散作者的写作精力。然而，我们也十分愿意向他们转达特殊的咨询。所有 Wrox 的作者都会为他们的书籍提供支持。他们将会把他们的回答使用电子邮件发送给客户和编辑，所有读者都会从中受益。

Wrox 的支持过程只为直接与我们的印刷书目内容有关的问题提供支持。我们的 http://p2p.wrox.com/论坛公共邮件列表会为超出书籍正常支持范围的问题提供支持。

0.4.4 p2p.wrox.com

为了与作者和同仁进行讨论，可以加入 P2P 邮件列表。除了我们的一对一的支持系统之外，我们特有的系统还可以通过邮件列表、论坛和新闻组，提供 programmer to programmer(程序员与程序员)之间的联系。如果用户向 P2P 发出了一个查询，就可以相信我们的邮件列表中的许多 Wrox 作者和其他业界专家都会处理这个查询。在 p2p.wrox.com，用户可以找到大量的不同邮件列表，它们不仅会对用户阅读本书提供帮助，而且也会对用户开发自己的应用提供帮助。

特别适合本书的列表是 pro_vb、pro_vb_dotnet 和 vb_dotnet。

为了订阅邮件列表，可以遵循如下步骤：

1. 前往 http://p2p.wrox.com/
2. 从左面的菜单栏中选择合适的范围
3. 单击希望加入的邮件列表
4. 依照订阅指示，填充电子邮件地址和密码
5. 答复收到的确认电子邮件
6. 使用订阅管理器加入更多的列表，并且设置电子邮件选项

为什么这个系统可以提供最好的支持

用户可以选择加入邮件列表，或者也可以把它们作为每周的文摘进行接收。如果用户没有时间或者条件接收邮件列表，用户还可以搜索我们的在线文档。特有的 Lyris 系统会删除垃圾邮件，并且会保护用户的电子邮件地址。有关加入或者离开邮件列表，以及其他有关列表的常见问题，可以发信到 listsupport@p2p.wrox.com。

目　　录

第1章　基础知识—— 事务处理、数据访问以及应用体系结构

本章将会介绍事务处理，它们的的概念，以及它们的工作方式。我们将会讨论事务处理的4 个基本属性。而且，我们还会了解使用一个数据存储的简单事务处理，以及涉及处理两个或者多个数据库数据的复杂事务处理。

本章的第二部分将会涉及应用的体系结构，并且会介绍访问数据的技术。毕竟，为了处理数据，我们必须首先能够访问数据。我们还会了解已经演化成.NET Framework 的以前的体系结构和技术。

读者可能奇怪为什么必须要了解较老的应用体系结构和技术。答案很简单。尽管.NET Framework 与以前使用的体系结构和技术有所差异，然而，很多体系结构原理仍然符合.NET 处理问题的方式，所以，熟悉以前的体系结构和技术将会帮助读者避免在学习上走弯路。

本章奠定了本书其余部分的基础。本章介绍的大多数概念将会贯穿本书的其余部分。当我们学习完本书之后，我们将会发现.NET Framework 实际上是由许多类似于 ADO.NET、ASP.NET，以及新的公共语言运行时(Common Language Runtime (CLR))这样的技术构成的，我们可以将 CLR 看作是 COM+服务的替代者。就如我们将要在本书中学到的，通过将所有这些技术结合在一起，应用开发者就可以使用各种方式执行事务处理。由于.NET 提供了如此众多的新技术，所以我们有必要逐一述及它们，以便读者在设计和实现自己的事务处理型应用的时候，能够自如地决定最好的方法。

在本章结束之后，读者应该能够回答如下问题：

- 什么是事务处理？
- 事务处理的 ACID 是什么？
- 什么是分布式事务处理？
- 什么是两阶段提交协议？
- 为什么我们仍然需要 COM+进行分布式事务处理？
- .NET 提供了什么？

1.1　事务处理概述

对事务处理(transaction)这个单词最早的有记载的使用出现在大约 1460 年的罗马民法典(Roman civil law)中，它用于定义对纠纷的调节。transaction(事务处理)是从单词 transactionem

直接借代而来，这个词表示“协定或者完成”。200 年之后，或者更晚的时候，单词事务处理才演化成为反映了商务含义。

事实上，当人类开始在地球上出现的时候，事务处理就已经存在。在我们讨论事务处理的当前定义之前，让我们首先回溯到大约 2 百万年前，拜访一下虚构的 Grog 公司的早期创始人 Grog。Grog 的第一笔商务事务处理是和名为 Borg 的本地族人进行的，Borg 收集了大量 Grog 仰慕的珠子，与此同时，Borg 也仰慕 Grog 收集的贝壳。Grog 提议使用 5 个贝壳换取 Borg 5 个珠子。Borg 接受了，交易也就达成了。这个简单的物品交换就是一个事务处理。

Grog 如约给了 Borg 5 个贝壳。然而，Borg 决定看看如果拒绝付给 Grog 珠子将会怎样，因此终止了交易。接着，Grog 使用猛犸象牙揍了 Borg 的脑袋，然后抢回了贝壳，这样，事务处理就失败了。

尽管 Borg 头很疼，但是他还是一个聪明人，很快向 Grog 道了歉。这一次，Borg 给了 Grog 5 个珠子，换回了 5 个贝壳，事务处理就完成了。

现在我们已经看到了一个简单的事务处理，接下来我们将会回到现在，分析我们刚才看到的情景。

编程术语中的事务处理与我们从 Grog 和 Borg 那里得到的事务处理的概念非常相似。它们也适用于如今的日常事务，例如从商店购买物品，或者向银行账户转账。

注意：

事务处理是绑定在一起的过程的集合。如果这个集合中的任何过程出现了问题，事务处理都会失败，所有的过程都要恢复到它们的最初状态。然而，如果集合中的所有过程都成功，那么事务处理就会成功，就会提交所有的过程。

为了强化理解，我们来分析一个更现代的示例：

假定一个人来到了 ATM(自动柜员机)前，想要从储蓄账户中转账 500 美元到支票账户中。为了进行资金划转，必须要进行两项操作。必须从储蓄账户中借出(删除)500 美元，还必须将 500 美元贷入(存入)支票账户。听起来很容易，这样的转账随时都会发生。但是，如果在储蓄账户中没有足够的钱，将会出现什么情况呢？如果在储蓄账户中没有足够的资金，事务处理就会失败。

既然我们已经知道了事务处理是什么，接下来我们就要讨论每个事务处理必须满足的要求。

1.2 ACID 属性

我们已经将事务处理定义为一个动作，或者一系列动作，它们可以将系统从一个一致状态转换成另一个一致状态。所有事务处理都必须遵守如下的 4 个基本要求，它们合称为 ACID 属性：

- 原子性(Atomicity)
- 一致性(Consistency)
- 隔离性(Isolation)
- 持久性(Durability)

1.2.1　原子性

原子性是指事务处理的全或无的命题。一旦启动了事务处理，它或者提交，或者放弃。

事务处理要么完全执行，要么根本不执行。当事务处理满足这些要求的时候，就称其为原子(atomic)。如果事务处理终止，或者失败，所有的事情都要同时退回到事务处理启动之前的原始状态。例如，当 Borg 拒付 Grog 珠子的时候， Grog 就会取回他的贝壳。整个事务处理都会放弃。换句话说，一辱俱辱。当 Grog 和 Borg 按照协议交易的时候，事务处理就会完全执行。同样的情况也发生在我们的 ATM 示例中。如果在储蓄账户中有足够的钱(而且没有硬件故障!)，事务处理就会完成，否则，将会完全终止。因此，这些事务处理都展示了原子性——全或者无。

尽管在讨论事务处理的时候，通常首先进入脑海的就是全或者无的场景，但是重要的是要记住，原子性属性只是事务处理必须要满足的 4 个必须的 ACID 属性之一。牢记这一点，我们来分析一致性。

1.2.2　一致性

一致性意味着构成事务处理的所有进程都不能够违反环境所规定的商务规则或者语义。

用更简单的话来讲，Grog 的事务处理的商务逻辑需要与 Grog 执行的所有事务处理一致。这意味着当 Grog 同意使用 5 个珠子交换 5 个贝壳的时候，Grog 一定要接收到 5 个珠子。与此同时，他会给 Borg 5 个贝壳。Grog 时代的商务规则非常简单：“x 物”换“y 物”。尽管这个商务逻辑非常简单，它还是展示了即使在 2 百万年前，事务处理也要维持一致性。

一致性要求事务处理结果所做的改变不能够破坏数据的完整性。例如，在我们的 ATM 银行事务处理场景中，在从个人的储蓄账户转账 500 美元到他们的支票账户之后，两个账户都要反映这次转账。因为事务处理是从一个一致的状态转换成另一个一致的状态，所以事务处理中的单独的操作可能会让数据处于不一致的状态，但是一旦提交事务处理，事务处理就必须保证所有受到影响的数据源中数据的一致性和完整性。

另外，在我们的 ATM 银行场景中，在转账之前两个账户都处于一致状态。在转账期间，首先要将钱从储蓄账户中抽取出来，在这时，数据就处于不一致的状态。作为转账的组成部分已经抽取了 500 美金，但是还没有记入另一个账户。然而，事务处理还没有完成。相同的 500 美金随后会存入支票账户。这个操作会有效地完成事务处理，两个账户都会回到一致的状态。

要记住，事务处理不负责强制数据的完整性，而是负责在事务处理提交或者终止之后，让数据返回一致状态。理解数据完整性的规则，以及编写可能的代码来强制完整性的重担，通常要落在建立商务组件的开发者的肩上。所以，做好准备吧。

当多个并发用户访问和修改相同数据的时候，事务处理必须要维护数据的一致性和完整性，这个概念将会与下一个 ACID 属性密切相关，它就是隔离性。

1.2.3　隔离性

一个事务处理中执行的所有操作必须与其他事务处理执行的操作隔离。

这个属性有的时候也称为串行化(serializability)，这是因为为了防止一个事务处理的操作影响另一个事务处理的操作，必须对请求进行串行化或者顺序化，以便相同的数据一次只能够服务于一个请求。