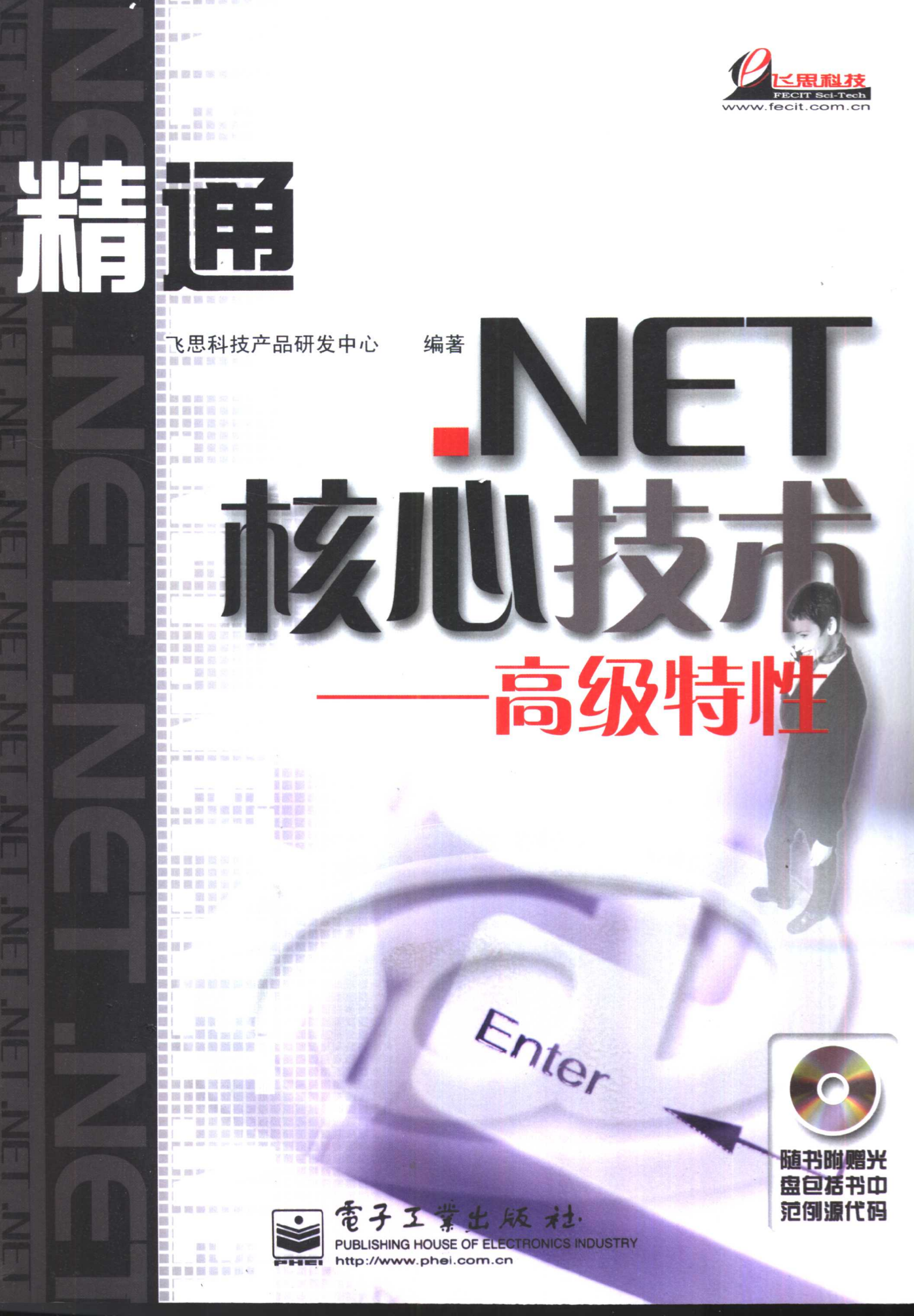
飞思科技
FECIT Sci-Tech
www.fecit.com.cn

精通

飞思科技产品研发中心　　编著

.NET 核心技术

——高级特性



随书附赠光盘包括书中范例源代码

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

精通 .NET 核心技术

——高级特性

飞思科技产品研发中心　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书是《精通.NET 核心技术——原理与构架》的继续，重于应用开发而非框架本身。主要介绍最实用的高级技术，包括远程处理、互操作、正则表达式、GDI+绘图、ADO.NET 等。远程处理为开发.NET 中的分布式应用提供了全面的解决方案。互操作提供了.NET 和传统 DLL、COM、COM+的互操作，它们为应用程序迁移到.NET 提供了完整的解决方案。GDI+是.NET 中新的二维绘图技术，进行图形图像处理的开发人员可能对该技术感兴趣。正则表达式是文本分析的利器，.NET 对此提供了强有力的支持。ADO.NET 是.NET 中的数据访问技术，利用它可以访问任何来源的数据，包括关系数据库、XML 数据源等。本书配套光盘包括书中完整的程序源代码。

本书面向广大程序设计人员，适合于作为学习和深入掌握.NET 技术的参考读物，指导中、高级技术人员进行开发工作。

图书在版编目（CIP）数据

精通.NET 核心技术——高级特性/飞思科技产品研发中心编著. —北京：电子工业出版社，2002.8
（精通系列）
ISBN 7-5053-7735-3

Ⅰ.精... Ⅱ.飞... Ⅲ.计算机网络—程序设计 Ⅳ.TP393

中国版本图书馆 CIP 数据核字（2002）第 043609 号

责任编辑：赵红梅 陆舒敏
印　　刷：北京大中印刷厂
出版发行：电子工业出版社　　http://www.phei.com.cn
　　北京海淀区万寿路 173 信箱　邮编：100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：56　字数：1433.6 千字　　附光盘 1 张
版　　次：2002 年 8 月第 1 版　　2002 年 8 月第 1 次印刷
印　　数：4000 册　　定价：79.00 元（含光盘）

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

出版说明

“精通”系列是电子工业出版社经典的技术精品丛书，一直受到广大读者，特别是计算机专业技术人员的关注。在这些专业人士的支持和鼓励下，“精通”系列已经成为一个航标，反映出目前国内外最新计算机技术的动态和发展方向。

“精通”系列中的每部著作完全是作者的呕心之作，代表了作者在该领域的最高成就，集成了作者多年的理论和实践经验，凸显了作者为计算机技术的发展做总结和展望的写作初衷。丛书的作者是由著名高校的学科带头人，国际上知名的教授、学者，权威的计算机专业人士和业界的集大成者组成的。他们的知识结构、理论和实践体系有着突出的卓越之处：

- 站在技术的最前沿
- 有最深刻的理论基础
- 实践环境具有广泛的代表性和权威性
- 结论的指导价值

因此，这样雄厚的写作班子保证了本系列丛书的高层次、高质量和高品质，也足以满足国内读者的高品位、高需求和高要求。除了作者之外，审校者同样实力超群，他们从理论的角度；读者需求的角度、技术成熟度的角度等不同的侧面，为作者提出了大量的参考意见和修改建议，使得每部著作的结构更坚实、体系更完整、理论更完善、针对性更强。

电子工业出版社计算机研发部本着服务于读者、服务于科技的精神，在选题上精益求精，综合考虑和平衡了目前技术上的热点、未来发展的重点以及潜在读者需求的卖点等多方面因素，精心推出“精通”系列，并将不断进行补充。

当然，我们的努力与读者的关怀是分不开的，衷心地欢迎读者提出意见和建议，促使我们推出更多、更好的精品书，以飨读者。

电子工业出版社计算机研发部

关 于 飞 思

新世纪之初的北京，一群满怀共同理想的年轻人聚集在飞思教育产品研发中心的旗帜下，他们将新的希望和活力注入了中国 IT 教育产品开发领域。飞思人在为把自己打造成为中国 IT 教育产品研发的精英团队而更加不懈努力。

21 世纪的今天，飞思人在多元化教育产品的开发和出版等方面已经迈出了坚实的第一步，开拓出属于自己的一片天空，初步赢来了涓涓细流。

如今，本着教育为科技服务的宗旨，飞思教育产品研发中心拓展为飞思科技产品研发中心，并以崭新的面貌等待您的支持与关注。

飞 思 人 理 念

我们经常感谢生活的慷慨，让我们这些原本并不同源的人得以同本，为了同一个梦想走到一起。

因为身处科技教育前沿，我们深感任重道远；因为伴随知识更新节奏，我们一刻不敢停歇。虽然我们年轻，但我们拥有：

“严谨、高效、协作”的团队精神

全方位、立体化的服务意识

实力雄厚的作者群和开发队伍

当然，最重要的是我们拥有：

恒久不变的理想和永不枯竭的激情和灵感

正因如此，我们敢于宣称：

飞思教育＝丰富的内容+完美的形式

这也是你和我共同精心培育的品牌 

的承诺。

“问渠哪得清如许，为有源头活水来”。路再远，终需用脚去量；风景再美，终需自然抚育。

年轻的飞思人愿为清风细雨、阳光晨露，滋润您发芽，成长；更甘当坚实的铺路石，为您铺就成功之路。

前言

关于.NET

.NET 是微软公司下一代的计算计划。其目标是把整个 Internet 整合为一个可计算的统一网络。

对开发人员而言，.NET 是一个完美的开发平台。它提供了一套公共的运行库，并制定了一套公共语言规范。所有符合该规范的语言都可以无缝使用这套运行库。在.NET 平台下，除了语法上的区别，各种编程语言没有本质的不同。它们共享公共类库，具有类似的编程模型和相差无几的功能。开发人员可以自由选择自己喜爱的语言开发程序。. NET 平台提供大量的服务，包括垃圾自动收集、面向对象的多线程、基于程序集的部署、异常处理、特性编程、远程处理、ASP.NET 网页框架、互操作、安全性等，使开发人员可以快速构架任何应用，从包括传统的桌面应用到面向 Web 的大型分布式应用。. NET 将彻底改变软件的开发方式、使用方式和发行方式，.NET 将是一场软件革命。

学习建议

.NET 是一个庞大的体系，它提供了多种编程语言，为开发各种.NET 应用提供了全面的、完善的服务。对.NET 了解得越深入，就越能体会到.NET 的优越性——随心所欲地挑选编程语言，轻轻松松地利用基础类库开发出最前沿的各种应用，如控制台应用、基于 WIN Form 的窗口应用、基于 Web Form 的 ASP.NET 应用、Web 服务、远程处理等。但.NET 这种强大的功能和无与伦比的优势往往让初学者感到困惑。下面的若干建议，可能对初学者有些帮助：

选择自己喜爱的编程语言。. NET 平台有非常多的编程语言供选择，但在.NET 平台下，不同的编程语言除了语法上有一些区别之外，功能上的区别将很小。它们共享.NET 框架的基础类库来实现各种应用。对已经有 Visual Basic 编程经验的人员来讲，可以考虑选择 Visual Basic.NET。因为已经获得经验，就可以帮助加快熟悉 Visual Basic.NET 的语法。如果已有 Java/C++ 编程经验，可以考虑选择 C#（读做 C Sharp），因为 C#的语法跟 Java/C++的很相像。本书的绝大部分例子，都提供了 Visual Basic 和 C#的两种实现，读者可以根据选用的编程语言阅读合适的版本。

熟悉命令行工具。精心设计的 Visual Studio.NET 集成开发环境使用户能够快速构建.NET 应用。但对初学者而言，最重要的可能不是如何高效地开发应用，而是明了集成开发工具做了哪些工作以及是如何做的。这时，最好的办法就是用.NET 的命令行工具手工实现相关任务。其实最重要的命令行工具就是编译命令。每种语言都有一个编译命令，例如，Visual Basic.NET 的编译命令是 vbc.exe，C#的编译命令是 csc.exe 等，它负责将特定语言编写的代码编译成程序集。其他比较重要的命令行工具还有 sn.exe（制作强名程序集时要用到）、AL.EXE（将模块装配成程序集）等。为了增强学习效果，读者可以先用集成开发工具实现，然后用命令行工具再做一遍。例如，本书所举的例子代码，笔者一般是在集成开发环境下创建

并编译通过的，然后经过适当修改，用命令行再编译执行。

重点掌握程序集和应用域。笔者认为，对程序设计师理解.NET 框架而言，程序集和应用域处于核心地位。掌握.NET 的第一个层次是了解.NET 本身，包括其框架构成、公共类型、框架类库等，特别要注意了解那些最常用的类型及其操作，例如字符串、数组、集合等。掌握.NET 的第二个层次就是理解.NET 中的核心元素，包括程序集、应用域、线程、命名空间等。而程序集和应用域则是这一阶段的重点和难点。第三个层次就是掌握.NET 框架提供的一些核心服务，如跨语言编程、异常处理、安全性等。掌握.NET 的第四个层次就是熟悉.NET 的高级特性，例如远程处理框架、互操作服务、ADO.NET 数据访问、GDI+绘图等。而第二层次的学习效果将直接影响到读者对.NET 的理解程度。读者在可以判断自己对.NET 的学习达到了何种层次后，从本套书中选择合适的内容。

关于本套书

本套书分为《精通.NET 核心技术——原理与构架》与《精通.NET 核心技术——高级特性》两册，涵盖了.NET 核心技术的各个层面，为开发人员提供了完整的知识架构，无论是开发何种.NET 应用的开发人员，都可以在这套书中找到翔实的技术参考。一旦通晓本书所介绍的内容，读者自会有 “会当凌绝顶，一览众山小”的感觉。

本套书具有以下特色：

- 迎难而上　本套书披露的内容，是.NET 的精华所在，但同时又是一块难啃的硬骨头，掌握起来有相当的难度。本套书反映了作者对.NET 技术的艰难摸索的过程，大多数是对某些问题深度思考后豁然开朗所得到的启示的总结。自从去年 3 月，作者就开始追踪.NET 技术，在摸索的过程中付出了相当多的热情和精力，逐渐领悟了.NET 的精髓。在艰难的探索过程中，有问题未解时的困惑，更有解开疑团后的云淡风轻。
- 循序渐进　.NET 所涉及的内容非常丰富，面对众多的新概念、新技术，不少读者会有茫茫然不知从何入手的感觉。本套书精心安排了章节顺序，以尽可能地符合.NET 学习曲线。对于新手，遵循本书安排的阅读顺序将把学习障碍减到最小。
- 手工打造　集成开发环境能简化应用程序的开发，但也会使开发人员不去思考简化背后的事情，从而妨碍对某些关键技术的理解。相反，通过命令行工具进行程序开发，包括编写源代码、编译等，程序员可能要做更多的工作，但是它更灵活。就学习一种技术而言，这种方式是可取的。本书穿插讲解了大量的.NET 命令行工具的用法，通过本书的学习，读者可以掌握常用的命令行工具。另外，为了彻底帮助大家理解某些“技术内幕”，本套书还提供了若干技术框架的手工实现。在清晰的论证和详细的实例分析下，读者的一切疑惑将一扫而光。

关于本书

本书是《精通.NET 核心技术——原理与构架》的继续，重于应用开发而非框架本身。主要介绍最实用的高级技术，包括远程处理、互操作、正则表达式、GDI+绘图、ADO.NET

等。远程处理为开发.NET 中的分布式应用提供了全面的解决方案。互操作提供了.NET 和传统 DLL、COM、COM+的互操作，它们为应用程序迁移到.NET 提供了完整的解决方案。GDI+是.NET 中新的二维绘图技术，进行图形图像处理的开发人员可能对该技术感兴趣。正则表达式是文本分析的利器，.NET 对此提供了强有力的支持。ADO.NET 是.NET 中的数据访问技术，利用它可以访问任何来源的数据，包括关系数据库、XML 数据源等。本书配套光盘提供书中完整的程序源代码。

本书面向广大程序设计人员，适合于作为学习和深入掌握.NET 技术的参考读物，指导中、高级技术人员进行开发工作。

本书由飞思科技产品研发中心策划并组织编写，刘晓华主笔，同时李华、李书德、张宏伟、薛德军、李国志、肖云、卢红娜、林军、万军等人也参加了本书的编写工作。杨艳女士认真阅读了本书的初稿，并提出了部分重要的参考意见。郭晶女士对本书的内容选材提出了很好的建议。李志强、赵红梅、陆舒敏等。对本书做了大量的文字处理工作，也使本书增色不少，在此表示衷心的感谢。

写作过程中参考了微软公司的相关资料，在此特做说明。由于水平有限，加之时间仓促，书中不足之处难免，敬请读者批评指正。

我们的联系方式：

电　话：　（010）68134545　68134811

E-mail：　support@fecit.com.cn

网　址：　http://www.fecit.com.cn　http://www.fecit.net

通用网址：　计算机图书、FECIT、飞思、飞思科技、飞思教育

飞思科技产品研发中心

目录

第 1 章　远程处理初步

远程处理是.NET 框架提供的一项强大的技术，利用它可以使位于任何位置的应用程序互相通信，这些应用程序可能在同一台计算机上运行，也可能位于同一局域网中的不同计算机上，或者位于相隔万里的差异巨大的网络中。

本章主要包括如下内容：

- 远程处理的基本概念
- 远程处理的过程
- 远程处理中涉及的对象及其关系
- 远程对象的生存期控制
- 远程处理的步骤
- 用编程或文件配置远程处理
- 用 IIS 发布远程对象

1.1　远程处理概述

.NET 远程处理本质上是一种进程间的通信的实现方法。它提供了实现任意数量的全面通信方案的工具。

1.1.1　功能及其优势

利用远程处理，我们可以进行以下处理：

- 在任意类型的应用程序域中发布或使用服务，无论该域是控制台应用程序、Windows 窗体、Internet 信息服务（IIS）、XML Web Services 还是 Windows 服务。
- 在二进制格式的通信中保持完整的托管代码类型系统保真度。

XML Web Services 使用 SOAP 格式化，这种格式化不会保持所有类型的详细信息。

- 通过引用传递对象返回到特定应用程序域中的特定对象。
- 直接控制激活特性和对象生存期。
- 实现和使用第三方信道或协议来扩展通信以满足特定要求。
- 直接参与通信进程以创建所需的功能。

.NET 的远程处理服务提供了进程间通信的抽象方法，它大大简化了分布式对象的访

问。在绝大部分情况下，服务程序通过简单的设置就可以把本地对象变成可以为远程提供服务的远程对象；而客户端则可以以类似访问本地对象的方法透明地访问远程对象。

然而，远程处理的真正优点在于它能够使位于不同应用程序域或者进程（它们使用不同的传输协议、序列化格式、对象生存期方案和对象创建模式）中的对象互相通信。更进一步讲，远程处理允许开发人员干预通信进程的任何阶段。

1.1.2 可远程处理的对象

根据分布式应用程序的用途，有两种简单的对象类别：可远程处理的对象和不可远程处理的对象。不可远程处理的对象不向系统提供复制它们或在其他应用程序中表示它们的任何方法。因此，这些对象仅可以从它们的原始应用程序域中访问。可远程处理的对象既可以使用代理在其应用程序域或上下文外部访问，也可以复制它们并且可以将这些副本传递到它们的应用程序域或上下文外；换句话说，某些可远程处理的对象通过值传递，而另一些通过引用传递。

1.1.2.1 按值传递对象

按值封送（MBV）对象声明它们的序列化规则（通过实现 ISerializable 来实现其自身的序列化，或者用特性 SerializableAttribute 标记，但是不扩展 MarshalByRefObject，远程处理系统创建这些对象的完整副本并将副本传递到进行调用的应用程序域。一旦副本到达调用方的应用程序域内，对它的调用就是对该副本的直接调用。而且，当 MBV 对象作为参数传递时，也是通过值传递的。除声明 SerializableAttribute 或实现 ISerializable 之外，无需做其他任何事情，就可以将类的实例跨应用程序或上下文边界按值传递。

当由于性能或处理原因将对象的完整状态和任何可执行功能移动到目标应用程序域有意义时，应当使用 MBV 对象。在许多方案中，这减少了跨网络、进程和应用程序域边界的冗长而耗费资源的往返行程。MBV 对象还可以从对象的原始应用程序域内直接使用。在这种情况下，由于不进行任何封送处理，因此，不创建任何副本而且访问非常高效。

另一方面，如果发布的对象非常大，那么在繁忙的网络上传递整个副本对于应用程序来说可能不是最佳的选择。此外，永远不会将对复制对象的状态所做的更改传回给起始应用程序域中的原始对象。在抽象级别上，这种方案类似于客户端浏览器所请求的静态 HTML 页的方案。服务器复制文件，将其写入到流中，发送出去，然后忘掉它。所有后续的请求都只是对其他副本的其他请求。

远程处理系统广泛使用可序列化的对象。对其他应用程序域中的对象的引用（由 ObjRef 类在远程处理系统中表示）本身是可序列化的；必须能够将它精确地复制并将副本发送给请求。同样，对于实现 IMessage 的消息对象也是如此，这是因为它们是调用信息和所有其他所需对象引用的一般容器。另外，仅传输数据的对象通常是 MBV 对象。例如，DataSet 扩展实现 ISerializable 的 MarshalByValueComponent。

1.1.2.2 按引用传递的对象

按引用封送（MBR）的对象是扩展 System.MarshalByRefObject 的可远程处理的对象。

根据已声明的激活类型，当客户端在自己的应用程序域中创建 MBR 对象的实例时，.NET 远程处理基础结构在调用方的应用程序域中创建表示该 MBR 对象的代理对象，并向调用方返回对此代理的引用。然后客户端将在该代理上进行调用；远程处理封送这些调用，将它们发送回起始应用程序域，并在实际对象上执行该调用。

如果客户端位于与 MBR 对象相同的应用程序域中，远程处理基础结构将向客户端返回对该 MBR 对象的直接引用，从而避免封送处理的系统开销。

如果 MarshalByRefObject 作为参数传入，当调用到达时，它变成另一个应用程序域中的代理。MBR 返回值，并且 out 参数以相同的方式工作。

当对象的状态和任何可执行的功能处在创建它的应用程序域中时，应当使用 MBR 对象。例如，具有内部字段且该内部字段是操作系统句柄的对象应扩展 MarshalByRefObject，这是因为操作系统句柄在其他进程中或其他计算机上的其他应用程序域中是无意义的。有时对象可能大得难以想像；对于功能强大的服务器还行，但通过网络发送到 33.6Kb/s 的调制解调器就不行了。

上下文绑定对象是一类更特殊的按引用传递的可远程处理的对象。它是从 System.ContextBoundObject（它本身从 System.MarshalByRefObject 继承）继承的 MBR 对象。可以将上下文当做应用程序域的子部分，它在执行期间为驻留在其中的对象提供某种资源充足的环境（例如保证对象不会被多个线程同时访问）。每个应用程序域都具有默认的上下文，大多数托管代码使用应用程序域的默认上下文创建对象，并直接从同一应用程序域内部调用成员，而不会产生与上下文有关的问题。所有从 ContextBoundObject 继承的类型都作为代理向其他上下文公开，即使它们位于同一应用程序域也是如此。

例如，假定有一个某类型上的方法，该类型是事务的一部分，因而受到特定于在其中创建它的上下文的规则的约束。应当使该类型从 ContextBoundObject 继承，这样就可以从对象本身的上下文访问该对象，而且系统可以强制实施有关与对象及其方法关联的事务的规则。如果从同一应用程序域内部的另一上下文中调用 ContextBoundObject，则将为调用方创建代理，但这种情况下，上下文间的通信没有通过信道系统，从而提高了调用效率。

考虑要远程处理哪种类别的类型时，请决定需要通过哪些边界。特定于某个特定上下文的对象只能从该上下文中直接访问。对于特定于某个特定应用程序域的对象也是如此。若要远程处理这两者中的任一种对象，在从服务器对象所特定于的边界内调用该服务器对象之前，远程处理系统必须成功通过上下文边界、应用程序边界或是成功通过这两种边界。由于通过每一边界都要花费处理时间，因而为了决定服务器应当是何种类型的可远程处理对象，就需要确定对象必须通过哪些边界。如果无需上下文检查就调用对象，则不应让远程类型扩展 ContextBoundObject（MarshalByRefObject 将执行得更好）。如果确实需要上下文检查，则应当扩展 ContextBoundObject，但是需要明白，在对象上进行调用之前，必须通过附加的边界。

1.1.3　远程处理的过程

远程处理的一般过程如图 1-1 所示，其中服务器对象是可以远程处理的对象。信道是