精通 Visual Basic .NET

网络与输入/输出技术

吕文达 编著

利用Visual Basic .NET程序语言探讨在.NET平台进行网络以及I/O相关技术的应用程序开发，本书涵盖的主要内容如下：

- 基础I/O与网络数据流概念
- 命名空间System.IO与数据流类探讨
- 命名空间System.Net与System.Net.Socket类的说明
- 文件与内存数据存取探讨
- 文本数据存取与加密数据流探讨
- I/O数据流与网络技术的综合运用
- OSI模型架构与网络通信协议概观
- Socket网络技术探讨与实现
- TCP、UDP通信协议以及HTTP、SMTP类的实现探讨
- POP与SMTP邮件软件范例实现
- FTP文件传输软件范例实现
- 探讨多播与消息交互范例的实现
- 远程服务机制与对象序列化技术

清华大学出版社

精通 Visual Basic .NET
网络与输入/输出技术

吕文达　编著

清华大学出版社

北　京

内 容 简 介

本书利用 Visual Basic .NET 语言探讨在.NET 平台进行网络以及 I/O 相关技术的应用程序开发。全书共分 10 章，主要内容包括：网络与 I/O 数据流、文件与目录操作、输入/输出数据流、文本处理与数据加密、.NET 网络程序设计、要求/响应模型与 HTTP 通信协议、TCP 与 Socket 应用程序、电子邮件与 FTP、UDP 通信协议与多播、远程服务与对象序列化等。

本书的读者要求具有以下基础：了解 Visual Basic .NET 基础语法，具备利用 Visual Basic .NET 编写基础应用程序的能力，熟悉类库的应用，具备.NET Framework 多线程、事件委派等基础概念的相关知识。

本书适合那些想了解如何利用 Visual Basic .NET 进行 I/O 操作的程序开发人员，使用 Visual Basic .NET 开发 Socket 与 TCP 网络应用程序的人员，想全盘了解.NET Framework 对于 I/O 与网络技术相关支持的.NET 程序设计人员，想了解.NET Framework 类库、支持 I/O 以及网络技术相关类别的 Java 程序设计人员。

图书在版编目(CIP)数据

精通 Visual Basic .NET 网络与输入/输出技术/吕文达编著. —北京：清华大学出版社，2005.8

ISBN 7-302-11348-3

Ⅰ. 精… Ⅱ. 吕… Ⅲ. BASIC 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2005)第 075663 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
责任编辑：桑任松
封面设计：陈刘源
排版人员：房利萍
印 装 者：北京市清华园胶印厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：21.25　**字数**：504 千字
版　　次：2005 年 8 月第 1 版　2005 年 8 月第1 次印刷
书　　号：ISBN 7-302-11348-3/TP·7468
印　　数：1～4000
定　　价：32.00 元

前　言

由于.NET 的特性与设计理念，本书将一起探讨 I/O 技术与网络程序设计这两个相关的技术主题，读者很难在不了解 I/O 数据流的情形下，利用.NET 写出实用的网络应用程序，.NET 由数据流概念出发，统一所有与数据输入/输出操作相关的存取技术，其中甚至包含了数据的加密解密、序列化以及远程服务与网络数据传输等内容。

I/O 是程序设计最基础、最重要的技术，.NET 利用数据流概念处理各种类型的数据读写操作，相关功能由数据流类 Stream 提供的类成员所公开，几个派生于 Stream 类的子类，针对各种不同的数据源，完成实际数据读写所需的相关功能，本书从数据流开始，带领读者循序渐进，一步一步了解.NET Framework 对于 I/O 操作所提供的支持，同时针对 I/O 类的运用，附有详细完整的示范说明。

除了基础的文件型 I/O，.NET 同样以数据流概念处理跨越网络的数据存取操作，程序开发人员因此能够用与存取文件数据相同的方式，以网络数据流对象进行网络的传输操作，本书前半段在完成基础 I/O 数据流的说明之后，紧接着将以其为基础，进行.NET 网络技术的相关内容探讨。

无论如何，除了数据传输之外，实际网络程序还牵涉到其他各种复杂的应用技术，例如 Socket 概念、通信协议等，为了让读者能够具备完整的知识，本书后半段对于这些主题，一并提供相关的说明。

不可讳言，网络技术是近代计算机应用发展过程中最重要的里程碑，网络的出现对于计算机信息工业具有革命性的影响。相对地，从程序开发的角度而言，强大的网络系统，代表高度的技术学习门槛，网络应用程序的复杂度也远远超过了一般传统的应用程序。幸运的是，.NET 以面向对象与网络概念为开发基础，因此利用.NET 开发网络应用程序的时候，程序设计人员将会发现，编写网络程序与一般 I/O 技术并没有太大的分别，本书将会说明 Visual Basic .NET 如何运用类库，完成相关的程序设计。

作　者

目　录

第 1 章　网络与 I/O 数据流

输入(Input)与输出(Output)操作，一般简称为 I/O，为所有程序语言必备的基本功能之一，从最简单的主控台(Console)I/O、文字文件数据读写，到跨越网络的数据流处理、复杂的数据库系统操作，均属于不同类型的输入/输出操作；I/O 在各种知识领域的应用非常广泛，了解程序语言处理 I/O 的技术能力，也就成为开发人员最重要的课题。

.NET 运用面向对象技术，提供 I/O 操作所需的相关支持。不同类型的 I/O 功能被封装于各种类中，程序开发人员只需运用相关的类对象，就能够在自行开发的应用程序中，整合出所需的功能。不仅如此，对于网络功能，.NET 引用 I/O 数据流的概念，提供相当丰富的支持。

本章从最简单的主控台开始，说明 Visual Basic 如何利用.NET 提供的 I/O 功能，进行简单的 I/O 操作，同时介绍具备 I/O 功能的相关类、类所属的命名空间以及其间的继承架构关系等等，同时针对所有 I/O 类的基类 Stream，作相关的介绍说明，最后，本章对于类库中支持网络功能的相关类也进行了探讨。

1.1　I/O 类

输入与输出是程序语言最基本的功能，学习一种程序语言除了基础语法之外，I/O 技术是首先必须面对的重要课题，程序设计师无法在不了解 I/O 的情形下，写出任何可用的应用程序，甚至最简单的“Hello World”范例程序，都必须利用输出功能，将数据输出到屏幕上，不论是显示命令的主控台界面，或者是友善的图形化窗口界面。

.NET 提供相当丰富，具备 I/O 功能的预定义类，这些类可以让我们轻易的完成各种相关的 I/O 操作，对于这些类，这一节将会对其进行概括性的说明，同时探讨各种类所能封装的功能。

1.1.1　类架构

.NET 对于 I/O 技术的支持，根据数据存取类型，提供封装 I/O 功能的各种类，这些类主要位于命名空间 System.IO，其中包含了操作文件目录的 FileSystemInfo 类、读写字符类型数据的 TextReader 与 TextWriter、处理字节数据的 Stream 类与专门用以存取原型数据如整型数据、布尔型数据以及浮点型数据的 BinaryReader 以及 BinaryWriter 等等。

图 1.1 说明了 I/O 类的功能与继承架构。

I/O 类大致上可以将其归纳为两大类，其中一类专门用以操作文件目录，另一类则是用以处理各种类型数据的存取操作。

处理文件目录的类中，派生于 Object 的 Directory、File 以及 Path 类具备通用于一般文件目录的处理功能，而另一大类则是由 FileSystemInfo 类所派生的两个子类 FileInfo 与 DirectoryInfo 等，这两个类特别适于处理特定的文件目录。

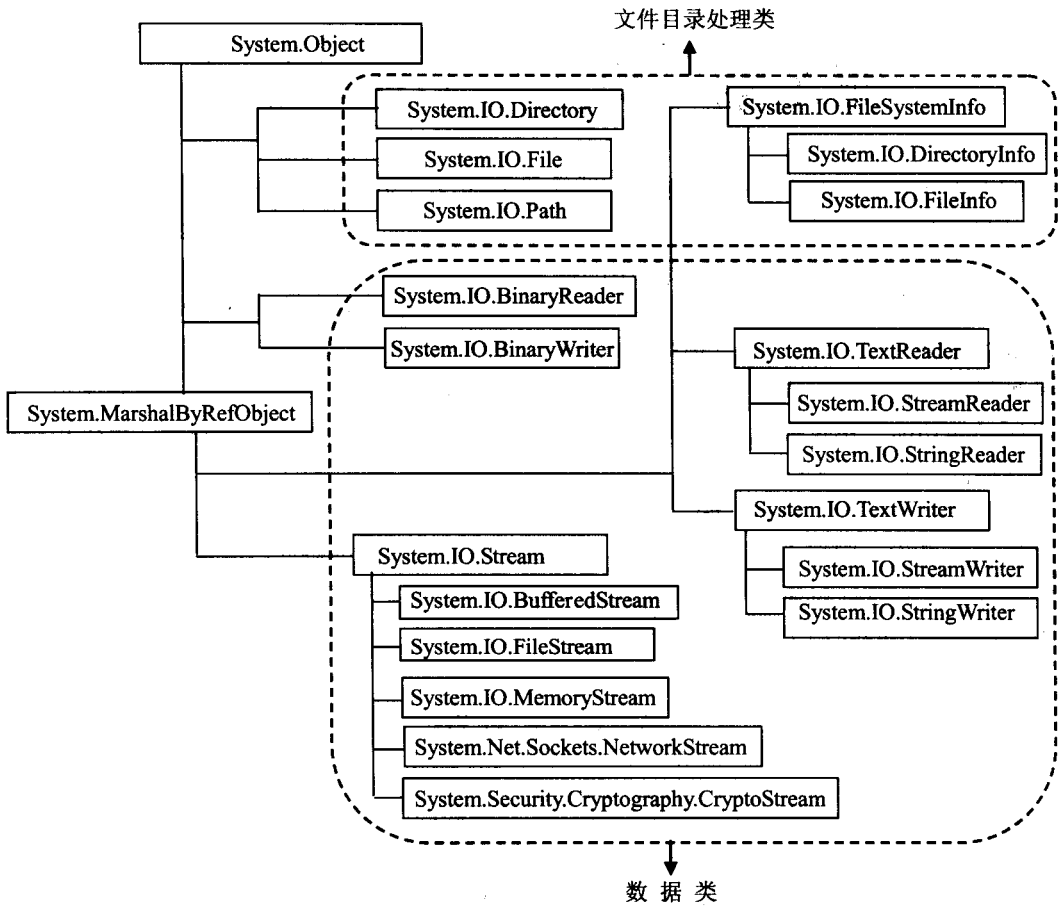

图 1.1

除了处理文件目录的类，其他的类均是使用于数据内容的读写操作，其中最重要且最基础的是 Stream 类，Stream 类是所有数据流类的基类，这个类派生的 5 个子类，BufferedStream、FileStream、MemoryStream、NetworkStream 以及 CryptoStream 等专门用以处理各种数据源的 I/O 操作，从类的名称可以很轻易的了解其中的差异，表 1.1 简要地列出这几个子类所能处理的数据源。

表 1.1 数据流类的几个派生子类及其功能

数据流类	功 能
BufferedStream	以内存缓冲区为数据源，进行数据的存取操作
FileStream	以文件内容为数据源，进行数据的存取操作
MemoryStream	以内存为数据存取的来源，进行数据的存取操作
NetworkStream	以连接网络为数据源，进行数据的存取操作
CryptoStream	支持数据的加密解密操作

数据流类及其派生的子类，提供了数据 I/O 操作所需的功能，这些功能甚至允许我们

对于数据进行加密解密的输出与输入操作，并且适用于各种不同的数据源。

除了文件目录系统及数据流，前述的类架构图中还有两组很重要的类，这两组类的主要功能在于存取不同格式的数据；数据的存取主要以三种内容格式来进行，二进制格式、字节格式以及文字数据格式等，上述的数据流类主要处理字节格式的数据，二进制格式的数据存取操作则由 BinaryReader 与 BinaryWriter 这两个类来进行，另外一组类 TextReader 与 TextWriter 则专门处理文字格式数据，表 1.2 根据所处理的数据格式，列举相关的类。

表 1.2　数据存取的格式及其对应的类

数据存取格式	对应的类
字节格式	Stream 以及其派生的子类
二进制格式	BinaryReader 以及 BinaryWriter
文字格式	TextReader、TextWriter 以及这个类下面派生的子类

表 1.2 中的各种类分别根据不同格式数据的存取操作，设计其所需的功能，这些类虽然独立存在，但是其间却必须相互合作，才能完成真正的数据读写操作。其中一种最重要的机制，便是数据流串接。采用这种技术，原始数据才可以以不同格式的方式进行读写，才不会干扰原始类对象的数据存储内容与存取过程。

1.1.2　串接数据流

尽管 I/O 类根据不同格式的数据存取具有不同的设计格式，然而真正可以连接底层数据源的，只有 Stream 类及其派生的子类，其他的几个类，无论读写二进制或是文字格式数据的数据流类，均必须串接数据流，以连接到底层的数据源，进行数据的存取操作。

图 1.2 以文件数据流为例，这个数据流被重新导向，串接至 StreamReader 对象，数据从字节类型转换成为字符类型进行读取，应用程序经过处理之后将其写入 StreamWriter 数据流对象，这些字符数据最后被导向 FileStream，以字节数据的类型，写入其连接的底层数据存储区。

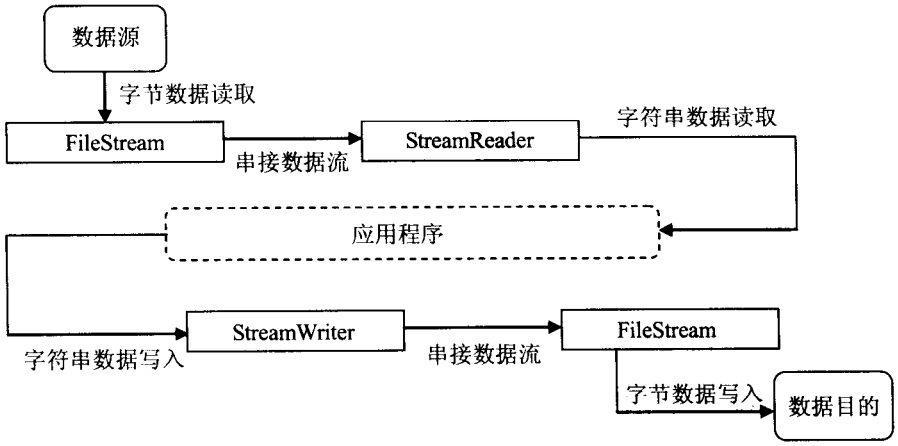

图 1.2

.NET 以面向对象的继承关系，组织所有的 IO 类，了解类之间的关联与继承架构，对于学习本书后续的 IO 主题，将会有很大的帮助。

1.2 文件目录操作

System.IO 命名空间设计了两组类用以进行文件目录的操作，分别是 Directory、File 类以及 FileInfo 与 DirectoryInfo 类，如图 1.3 所示。

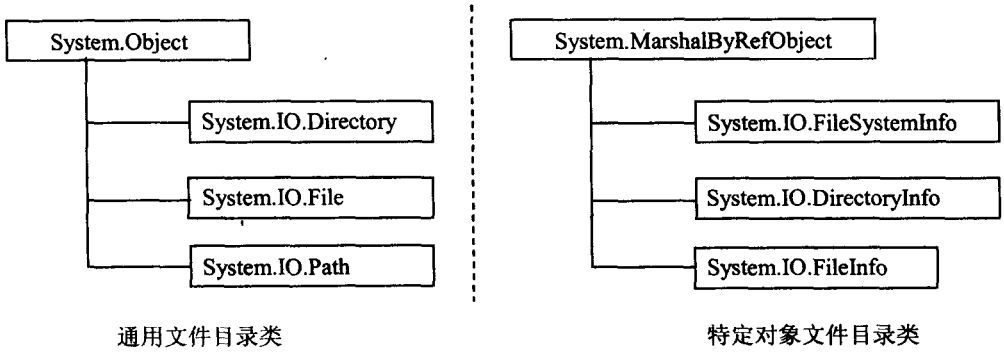

图 1.3

这两组类同时具备了操作文件目录，例如新建、打开、删除、复制以及移动文件或目录所需的功能。

其中最大的差异，在于图中左边派生于基类 Object，只提供共享的方法成员，这些成员并非针对特定的文件目录，因此不需新建特定的类对象，即可直接进行文件目录操作，而右边的类提供针对特定文件目录对象操作所需的方法及属性成员，利用这种类必须针对特定的文件目录，初始化并且新建其类对象，引用对象方法完成相关的操作。

其中必须注意的是，特定对象的文件目录类与通用的文件目录类不同，它主要由类 FileSystemInfo 提供文件目录所需的方法成员，然后再分别派生两个针对文件与目录的类 FileInfo 和 DirectoryInfo，本书下一章的内容，将会针对这几个用以处理文件与目录的相关类，进行详细的说明。

1.3 数 据 存 取

除了文件目录的相关操作，System.IO 命名空间里的其他类，提供各种形式数据的处理功能，这一节将以其读写数据功能，进行分类说明。

1.3.1 字节数据读写

这一组类设计用以存取字节数据，Stream 类为这一组类的核心基类，其下派生了几个重要的数据流子类，这些子类可以进一步分类归纳如图 1.4 所示，应用程序可以将其适当的整合，开发所需的功能。

图 1.4 中对于 Stream 几个类，作了一些分类。BufferedStream、FileStream 以及 MemoryStream，分别提供本机特定数据源的字节格式数据输入与输出相关操作功能。NetworkStream 则使用于跨越网络的数据存取操作。另外一个比较特殊的类 CryptoStream，则是使用于传送数据的过程中，输入/输出数据流的加密与解密操作。

我们同时可以注意到，BufferedStream、FileStream 以及 MemoryStream 位于 IO 命名空间，而 NetworkStream 则是位于 System.Net.Sockets.命名空间，CryptoStream 类则是另外一个命名空间 System.Security.Cryptography。

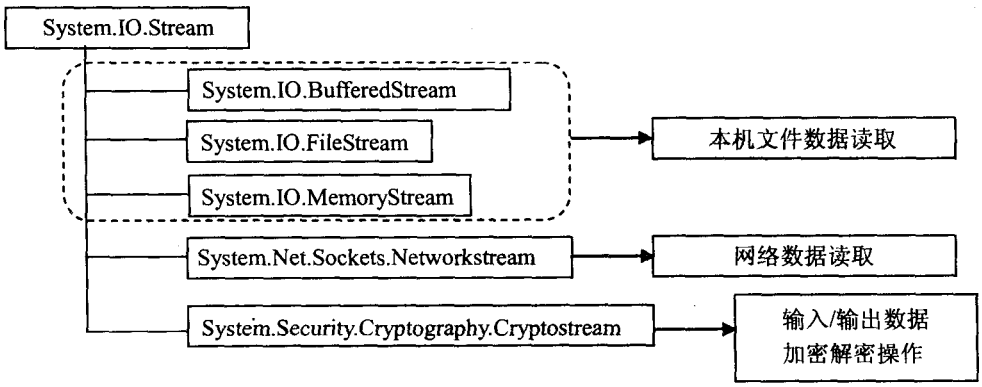

图 1.4

事实上，NetworkStream 与 Cryptography 亦非如前述的 FileStream 等三个数据流类，直接针对数据源新建其对象，NetworkStream 必须由连接网络端点的 Socket 对象返回其对象，而 CryptoStream 对象则是经过串接其他 Stream 类对象，新建其对象，以进行数据流的加密解密操作，其中加密解密的主题于文本处理一章，我们会有详细的探讨。

1.3.2　文字数据读写

文字类型的字符数据为另一种类型的数据流，提供字符组格式数据串流输入与输出功能类，主要由 TextReader 与 TextWriter 这两个类提供所需的功能，而根据所要读取的数据类型，字符与字符串，分别派生两个子类。StreamReader 与 StringReader 派生于 TextReader，而 StreamWriter 与 StringWriter 则派生于 TextWriter 类，相关的类组织结构如图 1.5 所示。

图 1.5 中将几个文字格式数据的读写类，依其格式功能作分类，左边为用以读取文字的类，其功能由基类 TextReader 定义，下面进一步根据所要读取的数据源，分别派生专属的子类，StreamReader 所定义读取数据流的文字数据，StringReader 则直接将一个字符串对象当作数据源进行读取；图右边则是写入数据所需的类，同样由一个基类 TextWriter 提供其定义的功能，并且进一步根据所要写入的数据源派生子类。

字符数据的读写类同样必须串接至特定的数据源，新建其对象，以进行文字数据的维护操作，其数据源有两种，一种是派生于 Stream 类的相关类对象，另外一种则是封装一个特定字符串对象，无论如何，不管哪一种类型的类，均是以 TextReader 与 TextWriter 类为其基类的延伸。

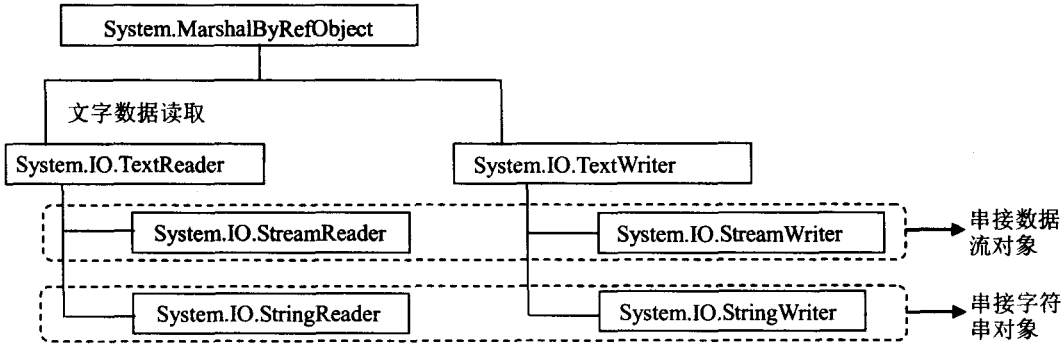

图 1.5

1.3.3 二进制数据读写

读写二进制格式数据的类主要包含了 BinaryReader 与 BinaryWriter，分别提供如布尔、整数、浮点数等基础数据类型的存取操作。

BinaryReader 与 BinaryWriter 串接原始的 Stream 类对象，并且以二进制格式读取字节的数据内容，也就是 Visual Basic 所定义的基本数据内容，如表 1.3 所列。

表 1.3 Visual Basic 定义的基本数据内容

Visual Basic 类型	数据长度(字节)	Visual Basic 类型	数据长度(字节)
Boolean	2	Double	8
Byte	1	Single	4
Char	2	Integer	4
Date	8	Long	8
Decimal	16	Short	2

下一章说明输入/输出数据流的课程内容，对于 BinaryReader 与 BinaryWriter 类，针对原数据的读写操作，会有完整的说明。

1.4 主控台 I/O

主控台是一种文字模式的 IO 接口，提供命令行形式的操作环境，其本身为一种 DOS 文字模式的窗口，看起来如图 1.6 所示。

命名空间 System.IO 所提供的 Console 类封装在命令模式下程序与系统沟通所需的 I/O 功能，例如从主控台读取数据的方法 Read、将数据写入主控台的方法 Write 等等。Console 类提供非常便利的方式，让用户完成系统的 I/O 操作。

虽然主控台 I/O 非常的简单，但是开发 Visual Basic .NET 的应用程序，对于 Console I/O 的应用并不常见，由于其贫乏的操作界面，我们几乎不会在实际的应用程序中使用它，然而，主控台却是一个示范 I/O 应用功能很好的起点，下面的内容使用 Console 类，新建一个与“Hello World”类似的应用程序。

图 1.6

1.4.1　一个简单的使用 Console 类的 I/O 应用程序

详细说明 Console 类的方法之前，我们首先示范一个文字模式的“Hello World”应用程序，此范例程序，提供一个文字接口，要求用户输入其姓名，然后根据用户姓名，输出适当的数据。

范例 1.1　ConsoleIO 程序

完整范例程序代码

```
Module Module1

    Sub Main()
        Dim strMessage As String

        Try
            Console.WriteLine("请输入姓名:")
            strMessage = Console.ReadLine()
            Console.WriteLine("HELLO " + strMessage)
            Console.ReadLine()

        Catch ex As Exception
        End Try
    End Sub

End Module
```

程序内容说明

以下这一行程序代码，等待读者写入数据：

```
strMessage = Console.ReadLine()
```

当用户将数据写入主控台，ReadLine 从主控台数据流读取下一行的字符数据，存到字

符串变量 strMessage，最后将取得的数据，加上“HELLO”字符，WriteLine 被引用，将一整行的数据写入主控台输出数据流，输出到画面上。

```
Console.WriteLine("HELLO " + strMessage)
```

程序一开始引用 Console 类的 WriteLine 方法，于画面上输出提示用户输入姓名的文字数据，如图 1.7 所示。

图 1.7

紧接着 ReadLine 方法被调用，当这个方法开始执行的时候，程序暂时停止，等待用户输入字符串数据，其执行结果如图 1.8 所示。

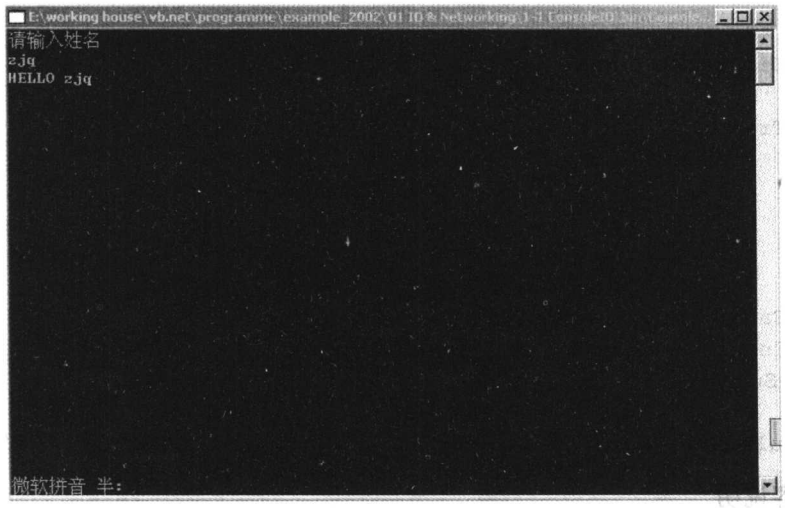

图 1.8

利用这两个方法，程序很轻易的便完成主控台的 I/O 操作，而除了这两个方法之外，你也可以使用 Read 与 Write 这两个读写数据的方法，进行单一字符数据的读写操作，稍后的内容对这些方法进行比较详细的说明。

上述的范例是一个很典型的 I/O 数据流操作，类 Console 从主控台输入数据读取用户写入主控台的数据，然后将数据写入输出数据流，最后输出到主控台，整个过程如图 1.9 所示。

图 1.9 中的数据由用户界面(主控台)连接到一个特定的数据源，Console.Read，这个数据源提供输入数据流所要读取的数据，输出数据流则是一个反向的操作，数据被写入输出数据流，最终输出到主控台界面。

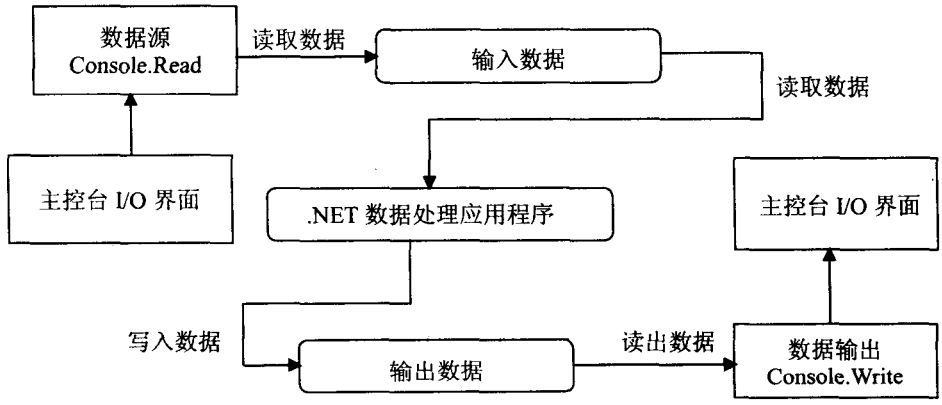

图 1.9

Console.Read 读取外部数据并且传输至输入数据流，Console.WriteLine 则扮演将数据写入输出数据流，并且传送到数据流的最终目的，事实上，对于.NET 而言，不仅仅是主控台，数据读取来源或是传送目的地，包含了其他所有的数据存储装置，一般最常见的便是文件系统，甚至于复杂的网络系统，都可以被当作输入/输出数据流的来源。

数据流是.NET 数据存取的标准化界面，无论上述的主控台、文件系统或是网络，除了连接的数据源不同，其数据的存取方式均是相同的，文件数据流与网络数据流分别为文件与网络等各种数据存储媒介的存取操作提供数据。

1.4.2 Console 类的方法成员

Console 类所提供的方法中，Read 与 ReadLine 从主控台输入数据流中读取数据，其中的 Read 方法从数据流目前位置读取下一个字符数据，而 ReadLine 则一次读取一整行的字符数据，也就是以一串字符加上一个断行符号或是直到字符结束为其一次读取的单位。

方法 Write 与 WriteLine 则是将指定的数据内容写入输出数据流，同样的，Write 一次将一个字符数据写入数据流，而 WriteLine 则是一次写入一整行的字符数据，其中行的定义与读取数据的方法 ReadLine 相同。

Read 与 ReadLine、Write 与 WriteLine 这两组方法相互成对，前者负责取数据，后者负责写入数据，必须注意的是，Read 与 ReadLine 这两个方法均只有一个版本，单纯的用以读取输入数据流的字符，而 Write 与 WriteLine 就没有这么简单了，为了将各种不同类型的数据顺利写入输出数据流，这两个方法分别针对不同的数据类型，提供了将近二十种重载的版本，例如将一个布尔值写入输出数据流，可以使用以下这个版本的方法：

```
Public Shared Sub Write( ByVal value As Boolean)
```