PROGRAMMER TO PROGRAMMER™

Visual Basic .NET Text Manipulation

String Handling and Regular Expressions Handbook

VB.NET

字符串和正则表达式参考手册

Francois Liger
Craig McQueen 著
Paul Wilton
康 博 译

清华大学出版社
http://www.tup.com.cn

VB.NET 字符串和正则表达式参考手册

Francois Liger
Craig McQueen 著
Paul Wilton

康 博 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：01-2002-3179

内 容 简 介

文本操作几乎存在于任何应用程序中，合理地处理文本可以提高应用程序的性能。

本书以大量的篇幅介绍了 VB.NET 中的文本处理，如文本的存储，文本的替换和追加等。书中详细地比较了各种文本操作方法的优劣和快慢，能帮助您在应用程序中从容选择使用最恰当的方法。本书用浅显的语言重点讲述了.NET Framework 处理字符串的最重要的工具之一——正则表达式，并介绍了如何构建自己的表达式用于匹配和操作文本。本书讲解实用生动，书中的大量代码可以直接用于您的应用程序中。

本书适合从事.NET 开发并想在应用程序中提高文本处理效率的开发人员阅读。

图书在版编目(CIP)数据

VB.NET 字符串和正则表达式参考手册/(美)尼格等著；康博译.
—北京：清华大学出版社，2002
书名原文：Visual Basic .NET Text Manipulation Handbook
ISBN 7-302-05943-8
Ⅰ.V… Ⅱ.①尼…②康… Ⅲ.BASIC 语言—程序设计 Ⅳ.TP312
中国版本图书馆 CIP 数据核字(2002)第 077172 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 http://www.tup.tsinghua.edu.cn
责任编辑：陈宗斌
印 刷 者：北京昌平环球印刷厂
发 行 者：新华书店总店北京发行所
开 本：787×1092 1/16 **印张**：15.25 **字数**：343 千字
版 次：2002 年 11 月第 1 版 2002 年 11 月第 1 次印刷
书 号：ISBN 7-302-05943-8/TP・3534
印 数：0001～4000
定 价：30.00 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。这套丛书讲述最新编程技术与开发环境的高级编程，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，每一本书皆出自软件大师之手。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。

前　言

按照 Wrox 的定义，文本操作是处理代码中的文本的过程。它不仅意味着在字符串中追加文本或替换文本，还包括在 String 类型中储存文本，以及在各种数据类型之间进行转换所使用的各种技术。创建的任何应用程序都有可能在执行过程的某个点处理文本，即使这种处理只是在一个 Console.Write()语句中将一个整数的字符串表示追加到文本中。.NET Framework 已经为开发人员提供了许多不同的数据类型和操作文本的方法。

本书主要讲述.NET Framework 如何处理文本，存储文本的不同方式，修改字符串的结果是什么，为什么需要一个单独的 StringBuilder 类，.NET Framework 把所有文本都存储在 Unicode 中的意义是什么。本书也讲解实现，同时解释各种可用的方法，比较它们的性能，以便读者在选择操作文本的方式时可以做出更好的选择。VB6 过去把所有的文本存储为变体，使用户几乎无法控制文本的处理方式，所以包含许多文本的 VB 应用程序经常在性能上大打折扣。Visual Basic.NET 改变了这一切。虽然不像在 C++中处理文本那样复杂，但通过面向对象的框架用户可以在代码中操作欲使用的文本。重要的是读者要理解.NET Framework 处理文本的方式。

可以使用的一个最重要的处理字符串的新类是 System.Text.RegularExpressions.Regex。这个类使所有的.NET Framework 开发人员能够使用正则表达式提供的强大的模式匹配语言。“匹配”的意思是看一个字符串是否遵守一定的模式以及是否包含我们预期的数据。通常这要比给一个方法传递字符串，然后捕捉一个异常(用无效数据更新数据库时发生的异常)更有效一些。语法的神秘一直是 VB 开发人员使用正则表达式的一个障碍，这些语法对普通人来说和要匹配的文本似乎没有什么关系。但在本书的第 5 到第 7 章，我们将指导您学习正则表达式的语言，解释每个组成部分的含义，并教您如何构建复杂的表达式以匹配文本中更复杂的模式。

没有正则表达式，即使是检查是否正确输入了电子邮件地址这样简单的任务也会耗费很多行代码。如果假设电子邮件地址已经存放在 myInput 变量中，可以用下面这一行代码生成一个布尔值，指明是否进行了成功的匹配：

```
bIsMatch = Regex.IsMatch(myInput, _
  "^[a-z](\w|\d|[-.])+@([a-z\d-]+\.)+[a-z]{2,4}$", _
  RegexOptions.IgnoreCase)
```

这看起来非常深奥，但您读完本书后，会发现它非常简明。为了简便，这个表达式没有对 myInput 变量做足够的限制(第 7 章会这样做的)，但它确实指定了下面的模式：

地址必须以字母开始，后面跟一个或多个字母，下划线，数字，连字符或句点，也可以是它们的组合，然后是“@”，它后面是一个字母的一个或多个模式，再后面是一个或多个字母、数字或连字符的可能重复的模式(后跟一个句点)，并以一个 2 到 4 个字母的模式结束。support@wrox.com 会匹配，但 support@wrox 不会匹配。

如果您有兴趣了解如何将该表达式语言用于 Visual Basic .NET 代码，那么本书的作者会带领您学习这种模式匹配语言，因为他们对该语言和 VB 本身都有丰富的经验。

本书读者

本系列的所有图书都以使用 Visual Basic .NET 进行开发的人员为对象，他们需要学习更多的东西才能完成具体的任务。本书也是如此，任何在应用程序中使用文本的开发人员(每个开发人员都会使用文本)都是本书的适用对象。本书代码的运行环境为 Visual Basic .NET Standard Edition 或更高的版本，但以前使用的代码和概念只适用于并只运行于.NET Framework SDK。

内容提要

本书讲解了处理文本的各种方法。下面是各章内容的介绍。

第 1 章　系统处理文本的方式

本章解释.NET Framework 处理文本的方式。它讲述了各种较基础的内容，如字符串如何在内存中存储。在解释了文本的存储方式后，读者会明白为什么考虑处理文本的方法是很重要的。本章会介绍一些 MSIL，但基本上是关于文本的概述，说明.NET 中内存的处理和管理(因此还有字符串占用内存的持续时间)。

第 2 章　String 和 StringBuilder 类

本章将详述 String 类，解释.NET 中如何实现字符串。还将讲解 StringBuilder 类的工作原理并比较这两个类的不同方法，了解执行某种操作的最快的方法是什么。学习 Microsoft 提供的共享源代码的 CLI 实现，以期了解不同方法快慢的原因，在什么环境下使用看起来慢一些的方法更合适。

第 3 章　字符串转换

本章讲述字符串和其他数据类型之间的转换所涉及的问题。讨论不同的区域如何以不同方式表示不同的数据类型，如数字，日期/时间，以及如何处理这些情况。我们还会探讨在数组和集合中存储字符串的问题，以及如何最好地操作以这种方式存储的字符串。

第 4 章　国际化

.NET Framework 中所有的文本都以 16 位的 Unicode 编码存放。本章对此进行了说明并告诉您面对现在的字符至少以 16 位存储这样的事实该采取什么措施。我们详细地讲述区域和针对不同区域表示文本的方法，还将讲解如何操作 Char 类型，因为在 Unicode 中，一个完整的字符可以由两个 16 位的 Char 类型组成。本章会告诉您如何在不同语言中显示文本。

第 5 章　正则表达式

本章为您介绍正则表达式匹配语言，如何使用正则表达式匹配文本模式。首先教读者如何完成使用 String 类的方法也能完成的相同工作，但您熟悉了语法后，我们会讲述该语言的一些独特的功能。我们还将带您创建一个 Regular Expression Tester，它是测试表达式的一个非常有用的工具。

第 6 章　正则表达式的高级概念

本章除了列举更多的表达式示例外，还将描述分组、替换和反向引用。分组允许表达式匹配成组的文本。这些组可以被捕获并赋予名称。反向引用可以用来提取前面的匹配，例如，可以为一个 HTML 结束标记匹配它对应的开始标记。

第 7 章　正则表达式模式

本章详细介绍各种正则表达式模式，它们用来匹配具体的数据类型，如成员，日期等。我们讲解这些模式的构成方式，因此能帮助读者构建自己的模式。其中一个模式是电子邮件地址验证程序，这是一个又长又复杂的表达式，用途很大，但经过讲解读者就会明白构建匹配许多文本模式的表达式是何等的简单。

附录

本书有 4 个附录。前两个附录用表格列出 String 和 StringBuilder 类的方法、属性和构造函数。第三个附录包含了很多正则表达式语法，选项和特殊的字符。最后一个附录是本书的支持信息和下载代码的信息。

下载代码

和 Wrox 的所有图书一样，本书包含的代码也可从 Wrox 的 Web 站点下载。但除了书中提到的技术以外，我还提供了一个程序集 Wrox.Text.dll，它包含了本书中很多有用的方法、属性和枚举，它们可以用于读者将来的工作中。通过把这些内容导入读者的应用程序，就可以立即使用本书讨论的许多有用的模式和方法。所有的类都包含在 Wrox.Text 命名空间中。

目　　录

第1章　系统处理文本的方式

文本是计算机应用程序中使用最为频繁的数据类型。有效地操作和使用文本对于构建强大和可维护的应用程序至关重要。强大和可维护的应用程序是运行顺利，节省内存，而且如果需要，还应是易于翻译的应用程序。但是如果使用不当，文本处理也可导致内存泄漏，性能下降，内存异常以及难于将应用程序导入到其他语言中。

为了帮助读者理解如何在 MS Windows 和.NET Framework 中使用文本，本章讲解以下问题：

- .NET Framework
- 作为数据类型的文本
- 字符和字符串
- 操作系统如何存储文本
- 典型的文本操作
- 常见的文本用法
- 国际化和本地化

本章介绍如何在典型的计算机操作系统中使用文本，具体讲解如何在 Visual Basic.NET 中使用文本。

本章给出了一些范例，介绍了在 Visual Basic.NET 和.NET Framework 之前执行文本处理任务的环境。本章中，术语“Visual Basic”指 Visual Basic 6.0 或之前的版本。在本章结束时，读者就可以理解计算机在处理文本时所面临的最基础性的问题。本章还介绍了.NET 如何处理字符串存储问题。本书的其他章节具体讲解了使用 Visual Basic.NET 处理文本的方法。

1.1　.NET Framework

为了理解系统如何处理文本，我们需要首先理解系统的各个组成部分。.NET Framework 是 Win32 架构的重要进步，因此.NET Framework 具有许多新概念。

.NET Framework 包括一个运行环境(也就是运行时)和一个类库。运行环境负责提供.NET 应用程序所需要的许多核心服务，其中包括内存分配和清除服务。因为字符串需要大量的内存管理工作，所以运行环境的实现将对使用字符串操作的应用程序的性能产生非常重要的影响。

1.1.1 公共语言运行时

运行环境通常称为公共语言基础架构(Common Language Infrastructure，CLI)。CLI 是 Microsoft 提交给 ECMA 的一项标准，而 ECMA 是负责处理信息和通信系统标准化工作的机构。其具体的标准引用号是 ECMA-335。

CLI 定义了可执行代码和运行环境的规范。运行环境也称为虚拟运行系统(Virtual Execution System，VES)。Microsoft 所实现的 CLI 被称为公共语言运行时(Common Language Runtime，CLR)。当讨论 Microsoft 专用的运行环境时，就使用 CLR；当讨论 CLR 所遵守的标准时，就使用 CLI。在本书中，大多数的讨论中使用了 CLR，因为 CLR 是我们所使用的 Microsoft 实现的运行环境。

CLI 的核心组成部分就是通用类型系统(Common Type System，CLS)。CTS 定义了在许多编程语言中所支持的类型和操作。我们在本书后面会详细讨论 CTS。

与 CTS 相关的是公共语言规范(Common Language Specification，CLS)，CLS 是提高语言互操作性的一组规则。.NET Framework 的主要目标就是允许开发人员以任何喜爱的语言从事开发，并统一在.NET 环境中运行这些程序。每一种语言都可以与其他语言进行交互。CLS 引导工具实现人员编写可以顺利过渡到其他语言的代码。

CLR 被认为是.NET 中编写的应用程序的“管理器”。CLR 确保应用程序符合安全规则，并向应用程序提供资源。在.NET 中编写的应用程序被称为托管代码，因为 CLR 管理了代码的运行方式。

托管代码通过公共中间语言(Common Intermediate Language，CIL)和文件格式存储和传输。CIL 是所有源代码语言都要被编译成的指令集。因此，如果您在 Visual Basic.NET，C#或任何其他的.NET 语言中开发，编译器都会将源代码转换为 CIL 指令。Microsoft 实现的 CIL 被称为 MSIL。

托管数据就是由 CLR 自动分配和释放的数据，这些数据存储在托管堆中。通过垃圾收集机制可以自动释放数据。

托管堆

CLR 提供了自动内存管理功能，它可以处理内存分配和再分配。对于开发人员来说这非常便利，因为在应用程序中最常见、也最难于跟踪的程序错误就是内存泄漏。当开发人员分配内存、但是忘记释放时，就会发生内存泄漏。当有些时候未确定何人负责释放内存时，也会出现内存泄漏。例如，如果一个组件分配了一些内存并把内存返回给调用者，那么谁来负责内存的释放，内存又应该什么时候释放呢？

因为必须分配字符串，所以内存管理的便利性就是使用字符串的便利性。

当开始启动应用程序时，CLR 就为应用程序预留一块内存。此内存块就是托管堆。所有的引用类型存储在堆中。稍后我们会讲到，String 是引用类型，因此它存储在堆中，

并和堆中其他分配的内存块一样得到处理。

图 1-1 演示了托管堆。

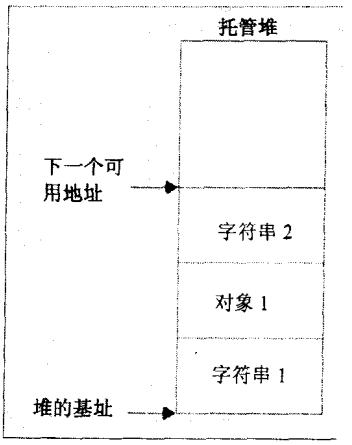

图　1-1

如图所示，堆具有带有内存基址的一块内存区。内存区是连续的，存储按线性方式分配。因为已经分配了堆内存，所以垃圾收集器在给系统返回分配的内存过程中只有一件工作，就是根据所要求的内存的大小分配下一个可用的地址。这可以快速分配内存。

内存分配非常简单，但是释放内存却要复杂得多。事实上，术语“垃圾收集”指释放不再使用的内存。计算机专家已经研究了执行垃圾收集的各种算法。.NET 文档简述了释放内存时垃圾收集器需要采取的步骤。在此我们就不再重复。

1.1.2　.NET Framework 类库

.NET Framework 类库是支持.NET Framework 编程的类型的集合。类型包括类、结构和基类型。例如，System.Text.StringBuilder 类和 Char 结构都是类型。类库包含的范围广泛，如用于 XML 处理，文件 I/O，绘图，线程，当然还有字符串操作的类。

在 System 命名空间中，有两个重要的类：

- Char——此结构是 Unicode 字符，具有转换字符和对字符分类(例如空白)的方法。
- String——此类是不变的字符串，具有操作字符串的方法。附录 A 是 String 类的构造函数、属性和方法的完整列表。

System.Text 命名空间包含重要的 StringBuilder 类，可以用于递增地构造字符串(附录 B 上具有 StringBuilder 类的构造函数、属性和方法)。命名空间也包含一些类，将一种字符编码转换为另一种，比如 Encoding，ASCIIEncoding 和 UnicodeEncoding 之间的转换。

System.Text.RegularExpressions 命名空间包含各种类，用于构造和执行正则表达式，例如 Capture，CaptureCollection，Regex 和 RegexCompilationInfo。

简而言之，下面就是本章要讨论的要点：

- .NET Framework 类提供的文本操作功能
- CLR 负责提供应用程序服务，包括分配和清除内存以支持字符串
- 在 CLR 中实现作为垃圾收集的内存管理功能

接下来，我们详细介绍数据类型。

1.2 文本是一种数据类型

在编程语言中，数据类型就是具有预定义特征的数据。数据类型定义的规则，规定了数据应该具有什么样的值。大多数编程语言具有内置的数据类型。数据类型通常包括整型，浮点型和字符串型。

强类型化的编程语言，比如.NET 语言，实施了这种规则。如果用户试图向一种数据类型赋予与之不对应的值，就会收到错误消息。通常这些规则由编译器进行检查，这样数据类型错误就可以在早期捕获，不致于使之进入运行时代码。

除了规定数据的值之外，对数据类型能执行的操作也有限制。例如，对字符串数据类型进行除法操作就没有意义。

文本是限定规则最少的数据类型。因为限定对文本可以携带的值的规则最少，所以当程序员无法确认数据源中值的范围(比如从外部数据源检索数据)时，就要使用文本数据类型。但是，由于文本数据无法有效存储，所以使用文本数据类型也会导致问题。

在某些编程语言中，文本数据类型非常难于处理。Visual C++就是一个例子。表 1-1 给出了当使用 Visual C++存储文本时可以选用的一些数据类型。

表 1-1 在 VC++中存储文本

选　项	范　例
字符数组	char myString[32];
宽字符数组	w_char myString[32];
标准 C++ 库	std::string myString;
MFC CString	CStr myString;
OLE Automation 字符串	BSTR myString;
BSTR 的类包装器	CComBSTR myString;

除了上表所列举的之外，还有在不同字符串类型之间进行转换和操作的函数。事实

上表示字符串并不需要这么多方法，在此所列举的众多方法只是历史上不断累积的产物。最初，字符存储在一个字节中。随着应用程序越来越变得国际化，在某些语言中就要求使用两个字节来表示所有的字符。

读者可能注意到了，我们在开始使用术语“字符串”表示文本。我不知道何人发明术语“字符串”来表示编程语言中的数据类型。但是，弄清楚这个术语是如何产生的却很容易。字符串是全称，表示字符的一个序列(或一串)。本章剩余部分我们使用术语字符串来代替术语文本。

因为.NET 是一种新的软件平台，所以我们有机会重新开始定义 String 数据类型，而避免其他编程语言中的混乱。

1.2.1　Visual Basic.NET 的数据类型

Visual Basic.NET 中的数据类型构建在 CTS 的基础之上。.NET Framework 的目标就是使程序员可以随意开发各种与.NET 兼容的语言，但是这些语言却在相同的基础平台上操作。CTS 定义了.NET 语言必须实现的数据类型。

1. 值类型和引用类型

.NET 中两种基本的数据类型是值类型和引用类型。

值类型包括整型，枚举，结构和字符。值类型直接存储在栈上，栈是操作系统分配的一个连续的内存区域，用于快速访问数据。CLR 对每一个正在运行的进程使用虚拟栈。因为值类型的容量是已知的，所以它可以存储在栈上。值类型可以直接访问，而从不通过引用访问。当复制值类型时，值就复制到另一个存储位置。

引用类型包括类，接口，数组和字符串。引用类型存储在托管堆中，托管堆是用于动态内存分配的内存区域，其中可以按任意次序分配和释放内存块。引用类型的容量直到运行时才能知道。这就是使用堆存储引用类型的原因。

可以通过对存储区的引用(类似于指针)访问引用类型。这使得垃圾收集器可以跟踪引用，当引用不存在时，就释放存储区。当复制引用变量时，实际上是复制引用(也就是内存地址)，而不是值本身。

2. 基本数据类型

.NET Framework 类库是与 CLR 紧密集成的类型的集合。类库是面向对象的，这表示所有的类型都有属性和方法。一些类型使用得非常普遍，因此为描述这些类型提供了快捷方式。这些类型被称为基本数据类型。

基本数据类型通过关键字识别，这些关键字就是 System 命名空间中预定义的类型的别名。基本数据类型与它的别名表示的结构类型没有区别：保留字 Char 就等于