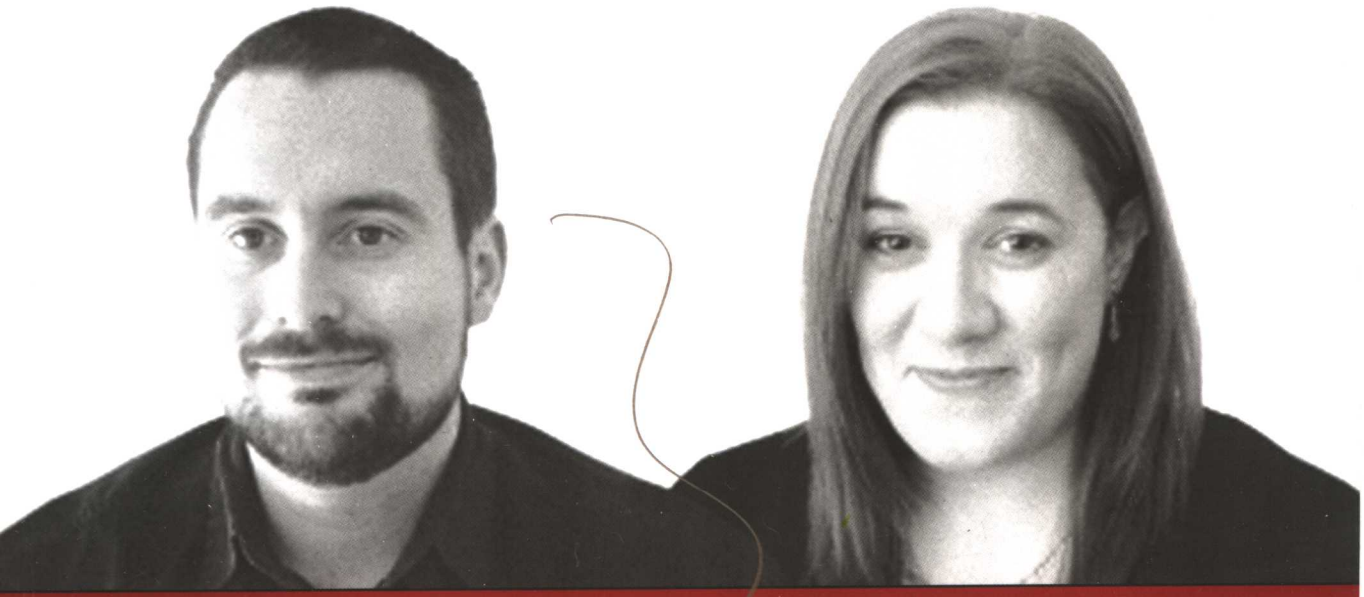
PROGRAMMER TO PROGRAMMER™



C# 2005 Programmer's Reference

C# 2005 编程进阶与参考手册

(美) Adrian Kingsley-Hughes 著
Kathie Kingsley-Hughes
施宏斌 译

wrox™
清华大学出版社

C# 2005 编程进阶与参考手册

(美) Adrian Kingsley-Hughes 著
Kathie Kingsley-Hughes
施宏斌 译

清华大学出版社
北 京

Adrian Kingsley-Hughes　Kathie Kingsley-Hughes
C# 2005 Programmer's Reference
EISBN：0-470-04641-4

北京市版权局著作权合同登记号　图字：　01-2006-6183

图书在版编目(CIP)数据

C# 2005 编程进阶与参考手册/(美)肯斯理-赫金斯(Kingsley-Hughes, A.)，(美)肯斯理-赫金斯(Kingsley-Hughes, K.)著；施宏斌 译. —北京：清华大学出版社，2007.11
书名原文：C# 2005 Programmer's Reference
ISBN 978-7-302-16380-0

Ⅰ.C… Ⅱ.①肯…②肯…③施… Ⅲ.C 语言—程序设计—技术手册 Ⅳ.TP312-62

中国版本图书馆 CIP 数据核字(2007)第 168761 号

责任编辑：王　军　徐燕萍
装帧设计：孔祥丰
责任校对：成凤进
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　**邮　　编**：100084
c—service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 刷 者：北京市昌平环球印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：23.25　**字　数**：566 千字
版　　次：2007 年 11 月第 1 版　　**印　　次**：2007 年 11 月第1 次印刷
印　　数：1～4000
定　　价：39.99 元

前　言

本书非常详细而全面地介绍了 C#程序设计语言。本书不是“5 分钟学习 C#”式的手册，也不是那种教您“照猫画虎”地创建一些与您的实际工作需要或兴趣毫无关系的应用程序，让您“知其然而不知其所以然”的书。那类书籍仅仅非常简单地介绍了一下程序设计语言。

1.1　本书特点

本书并不像其他那些简单概述 C#语言的书(那些书就像全球地图一样，仅提供了各大陆或各个国家的轮廓、或者湖的面貌等等，但没有更详细的信息)，本书将从各方面深入介绍 C#语言以及设计这些语言特性的目的(还是用地图来做比喻，您将从地球的范围拉近到街道的范围，您将能看到每个街道的名称和建筑物)。

但本书前面几章仍然会花一定的时间来概述 C#语言，比如：C#概念、如何开始使用 C#、C#语言基础。这几个基础性的章节为深入研究 C#各方面的特性提供了一个过渡。

在几个基础性的章节之后，本书将深入研究 C#。从介绍 C#语言基本概念和语言结构开始，本书将深入探讨 C#中的类型、变量和类型转换。在这些章节的基础之上，本书将仔细介绍 C#中的表达式和语句，随后介绍了如何在 C#中使用命名空间。

接下来本书讲述 C#中的类、结构、数组、枚举和委托，以及 C#中的异常、特性(attributes)、泛型(generics)和迭代器。本书正文的最后一章，介绍了 C#中安全代码和非安全代码的编码实践。最后，本书提供了一些附录，介绍 C#语法、命名规范、可移植性和 XML 文档注释等。

1.2　本书读者对象

本书不是 C#语言的入门书籍。本书面向那些已经对 C#有了基本了解的读者，可帮助他们进一步掌握和利用 C#的高级技术和语言方面的功能。如果您还没有 C#的任何经验，建议您先阅读 Wrox 系列图书中 C#语言入门级的书籍。入门级的书籍提供了 C#语言的基本知识，可作为掌握 C#高级技术的基础。

1.3　本书内容

从序言开始，本书包含以下内容：

- 第 1 章：C#简介。本章简要介绍 C#的概念、来源和历史。
- 第 2 章：开始使用 C#。开始使用 C#编程并不需要太多的软件支持。本章将介绍使用 C#编程所必需的软件支持，以及一些让工作更轻松的小技巧。
- 第 3 章：C#语言概述。C#是一种功能强大、非常灵活的程序设计语言，本章将带您快速纵览 C#，并对 C# 中最重要的语言特性做了强调。
- 第 4 章：C#的语言结构。本章介绍 C#的语言结构，着重介绍 C#的词法(Lexical grammer)、语法(syntactic grammer)、符号和指令。
- 第 5 章：C#基本概念。本章将介绍 C#中的关键概念，比如应用程序的启动和终止、成员及对成员的访问、重载等。
- 第 6 章：类型。从本章开始将介绍 C#语言的细节问题，首先是数据类型。本章将讨论值类型(value types)和引用类型(reference types)、以及装箱(boxing)和拆箱(unboxing)。
- 第 7 章：变量。本章将讨论的主题是数据操作的关键：变量。
- 第 8 章：转换。本章将讨论 C#中的类型转换，如特殊的类型转换、隐式类型转换、显式类型转换、标准的类型转换。
- 第 9 章：表达式。表达式是 C#编码的关键。本章将讨论 C#中的各类表达式。
- 第 10 章：语句。通常称一个或多个代码行为一个语句。本章将讨论 C#语句的结构并仔细分析 C#中各种不同的语句。
- 第 11 章：命名空间。本章将讨论 C#中如何使用命名空间，使用命名空间有助于消除二义性和解决命名冲突的问题。
- 第 12 章：类。本章将讨论类，以及如何在 C#中用类来组织代码。
- 第 13 章：结构。本章将讨论结构以及如何在 C#中使用结构。
- 第 14 章：数组。数组是一种重要的数据类型，数组以结构化的方式存储数据，并使之易于访问和使用。本章将讨论 C#所支持的各种类型的数组。
- 第 15 章：接口。 本章将详细讨论 C#中的接口类型。接口的声明、接口的成员、成员签名以及接口的实现。
- 第 16 章：枚举。枚举是一种强类型化的常量，本章将讨论枚举类型在 C#中的应用。
- 第 17 章：委托。本章将介绍 C#中委托的声明、实例化及调用。
- 第 18 章：异常。本章将详细说明 C#中异常的概念、异常的产生、处理和异常类。
- 第 19 章：特性。本章将介绍 C#中的“特性类”、特性的实例，以及保留的特性。
- 第 20 章：泛型。泛型是 C# 2005 中最有价值的新特性。本章将介绍如何声明和使用泛型类型。
- 第 21 章：迭代器。迭代器可以让代码更加精简。本章将详细讨论 C#中多种不同的迭代器。
- 第 22 章：非安全代码。本章讨论，在不危及项目安全的前提下，如何在 C#中充分使用非安全代码特性。
- 附录 A：C#语法。
- 附录 B：命名规范。
- 附录 C：标准库。

- 附录 D：可移值性。
- 附录 E：XML 文档注释。

1.4　选择合适的章节阅读

如何阅读本书完全取决与您。如果对 C#还不熟悉，最好先阅读本书第 1 章到第 10 章，然后您可以根据需要或 C#技能的提高选择其他章节阅读。如果您已经是一名 C#的开发者，本书更适合做为一本参考手册，而不是从头带尾逐章阅读，可根据需要选择部分章节阅读。

本书附录是有关 C#方面的资源和信息，仅供浏览。附录并不需要完全阅读，除非您非常热衷于 C#(当然您也可以完全阅读，但请注意我们的提醒)。

1.5　如何下载本书的示例代码

在读者学习本书中的示例时，可以手工输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从本书合作站点 http://www.wrox.com 上下载。登录到站点 http://www.wrox.com/，使用 Search 工具或使用书名列表就可以找到本书。接着单击本书页面上的 Download Code 链接，就可以获得所有的源代码。

注释：

许多图书的书名都很相似，所以通过 ISBN 查找本书是最简单的，本书的 ISBN 是 0-470-04641-4(2007 年 1 月，将使用 13 位数字的新的出版业标准 ISBN: 978-0-470-04641-8)。

在下载了代码后，只需用自己喜欢的解压缩软件对它进行解压缩即可。另外，也可以进入 http://www.wrox.com/dynamic/books/download.aspx 上的 Wrox 代码下载主页，查看本书和其他 Wrox 图书的所有代码。

1.6　勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但是事无完美，错误总是难于避免，如果您在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将非常感激。通过勘误表，可以让其他读者节省时间、避免阅读和学习受挫，当然，这还有助于我们提供更高质量的图书。

要在网站上找到本书的勘误表，可以登录 http://www.wrox.com，通过 Search 工具或书名列表查找本书，然后在本书的细目页面上，单击 Book Errata 链接。在这个页面上可以查看到 Wrox 编辑已提交和粘贴的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 www.wrox.com/misc-pages/booklist.shtml。请给 wkservice@tup.tsinghua.edu.cn 发电子邮件，我们就会检查您的信息，如果是正确的，我们将在本书的后续版本中采用。

1.7 p2p.wrox.com

P2P 邮件列表是为作者和读者之间的讨论而建立的。读者可以在 p2p.wrox.com 上加入 P2P 论坛。该论坛是一个基于 Web 的系统，用于传送与 Wrox 图书相关的信息和相关技术，与其他读者和技术用户交流。该论坛提供了订阅功能，当论坛上有新贴子时，会给您发送您选择的主题。Wrox 作者、编辑和其他业界专家和读者都会在这个论坛上进行讨论。在 http://p2p.wrox.com 上有许多不同的论坛，帮助读者阅读本书，在读者开发自己的应用程序时，也可以从这个论坛中获益。

要加入这个论坛，需执行下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读其内容，单击 Agree 按钮。

(3) 提供加入论坛所需的信息及愿意提供的可选信息，单击 Submit 按钮。

然后就可以收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

提示：

不加入 P2P 也可以阅读论坛上的信息，但只有加入论坛后，才能发送自己的信息。

加入论坛后，就可以发送新信息，回应其他用户的贴子。可以随时在 Web 上阅读信息。如果希望某个论坛给自己发送新信息，可以在论坛列表中单击该论坛对应的 Subscribe to this Forum 图标。

对于如何使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作原理，以及许多针对 P2P 和 Wrox 图书的常见问题解答。要阅读 FAQ，可以单击任意 P2P 页面上的 FAQ 链接。

目　录

第 1 章

C# 简 介

本章将简要介绍 C#语言，并回顾 C#的起源和历史。

1.1 C#的名称

根据 ECMA-334 的 C#语言规范(http://ecma-international.org/publications/standards/Ecma-334.htm)，C#的名称由一个大写的拉丁字母“C”(U+0043)，后加一个“#”号字符(U+0023)组成。C#读作 C sharp 或 see sharp。

C#这个名称的来源带有几分神秘。因为“#”字符由 4 个排列成方形的“+”号组成，所以一些人认为 Microsoft 选择“C#”这个名称，暗示了 C#是从 C++语言演化而来。另一种关于 C#名称的来源听起来更有道理，“C#”这个名称表明了“C#”同 C++语言的关系，就像 C 语言同 C++语言的关系一样，因为在 C++语言中，“++”表示变量的自增操作。在音乐中，“#”表示在音阶上“升半调”的音符，所以“C#”也可能表示在“C”这个读音上“升半调”。

爱好音乐的读者可能已经知道，计算机键盘上的“#”符号并不是音乐中表示“升调”的符号。在计算机中，“#”符号通常用来代表数字。使用“#”这个符号是因为在计算机的标准键盘上没有音乐中的“升调(U+266F)”符号，要打出音乐中的这个“升调(U+266F)”符号不太方便。尽管使用了“#”这个符号，这门语言也不能读做 see pound 或者 see hash，更不能读做 see gate!

1.2 C#概述

C#是 Microsoft 开发的一种面向对象的程序设计语言，也是 Microsoft 公司.NET 软件开发平台的重要组成部分。C#包含一系列称之为“类”的独立的程序设计单元，这些类可以相互作用和交互。

C#深深根植于 C++语言，同时毫无疑问地也受到了微软公司其他流行语言如 Visual Basic 的影响。C#一个最大的好处是它的语法(或者称为代码的结构)非常类似于其他流行的程序设计语言，尤其类似于 C++、Visual Basic、Java 和 Delphi，这使得拥有其他语言开发

背景的程序员通过最小的学习曲线就能迅速掌握 C#。并且，C#语言比 C++和 JAVA 都更加简洁。

1.2.1 C#的历史

C#的首席设计师是 Microsoft 的 Anders Hajlsberg。Hajlsberg 曾任职于 Borland 公司，20 世纪 80 年代早期，他是著名的 Pascal 编译器的创始人。1996 年，Hajlsberg 离开 Borland 加盟 Microsoft 公司，并把宝贵的经验带到了 Microsoft，曾领衔开发了 J++和 Windows Foundation Classes(Windows 基础类库，简写为 WFC)，随后又领衔开发了 C#和公共语言运行库(Common Language Runtime，简写为 CLR)。CLR 本质上就是虚拟机和运行库，是.NET 的基石(.NET Framework 允许代码不经过 CLR 直接在主机系统上运行)。Hajlsberg 对 C++、Delphi、Java 以及 Smalltalk 等语言中存在的瑕疵感到非常不满，并决心设计一门更加优秀的语言——C#。所以，C#同 C++、Delphi、Java 等有着许多类似之处。

1.2.2 C#与公共语言运行库

所有的.NET 程序都依赖于公共语言运行库，C#可以充分利用公共语言运行库的各种优势。用 C#编写的所有应用程序都运行于 CLR 之上(或者说运行于 Microsoft .NET Framework 之上)，就像 Visual Basic 应用程序需要相应的运行库才能运行一样。

关于.NET Framework 的相关信息以及相关下载，请访问 Microsoft 的网站 http://msdn.microsoft.com/netframework/。

CLR 的主要功能包括：

- **托管代码**。托管代码由 Visual Studio 产生，运行于.NET Framework 之上。
- **简单和自动的应用程序安装**。该功能由全局程序集缓存(Global Assembly Cache，简写为 GAC)实现。
- **内存管理**。CLR 为程序员提供了简单、高效的内存管理方式，程序员只需使用更少的代码，就能获得更好的性能。
- **自动垃圾回收**。当对象不再被使用时，.NET Framework 将自动释放内存。
- **执行期间良好的安全性**。.NET Framework 使用集成安全模型，只有在程序集中明确定义了允许的权限，才能访问资源。

1.2.3 迁移到.NET

您可以快速迁移到.NET。

Microsoft 的.NET 平台包含 4 个基本组成部分：

- .NET 服务构件模块(Building Block Services)，如 Passport
- .NET Framework 精简版(.NET Compact Framework)，用于支持移动电话和个人数字助理(PAD)等小型设备
- 通过 XML 集成实现的.NET 用户体验

- .NET 基础构架，比如.NET Framework 公共语言运行库、.NET Framework 类库(.NET Framework Class Libraries)、.NET 开发工具(如 Microsoft Visual Studio.NET)。

任何一种.NET 语言都集成了.NET Framework 类库。.NET Framework 类库支持文件I/O、数据库操作、XML(Extensible Markup Language，可扩展标记语言)、 SOAP(Simple Object Access Protocol，简单对象访问协议)等。

对于.NET 程序设计和.NET 软件开发来说，最重要的是利用起到支撑作用的.NET Framwork，因为它包括了运行时环境和丰富的类库。

1.2.4 相关标准

对于 C#来说，一件非常重要的事情就是 Microsoft 已经将 C#语言的编写标准提交了ECMA(European Computer Manufacturers Association，欧洲计算机厂商协会)。

2001 年 12 月，ECMA 发布了 ECMA-334 C#语言规范；在 2003 年，C#已经成为了 ISO 标准(ISO/IEC 23270)。

ECMA-334C# 语言规范可以从 ECMA 的网站免费下载，网址为http://www.ecma-international.org/publications/standards/Ecma-334.htm。

ISO/IEC 23270 标准可以从 ISO 网站购买(http://www.iso.org)，或者下载免费的电子版文档。

在 Visual Studio 2005 中，Microsoft 增加了对范型(generics)、分部类型(partial types)等特性的支持，并已经将这些新特性提请作为标准，但这些新特性尚未成为标准列在目前的规范之中。

1.2.5 其他实现

C#已经从一种 Microsoft 的专有语言发展成为了具有其他独立实现的开发项目，其中两个最大的独立实现项目是：

- DotGNU—http://www.dotgnu.org/
- Mono—http://www.gotmono.com/

围绕 C#建立的团体正走向繁荣，这为那些想充分利用 C#的程序员提供了更多的选择和灵活性。然而，对于任何一个独立实现项目，都与标准平台存在一定的差异。

1.3 C#代码的简单示例

C#代码的结构是怎样的？本书稍后将展示大量的 C#代码，但现在仅给出一个简单的“Hello，World!”程序的代码示例：

```
public class MyClass
{
   public static void Main()
```

```
    {
        System.Console.WriteLine("Hello,  World!");
    }
}
```

这段代码在编译后，将在屏幕上输出一段文本“Hello，World!”(如图 1-1 所示)。

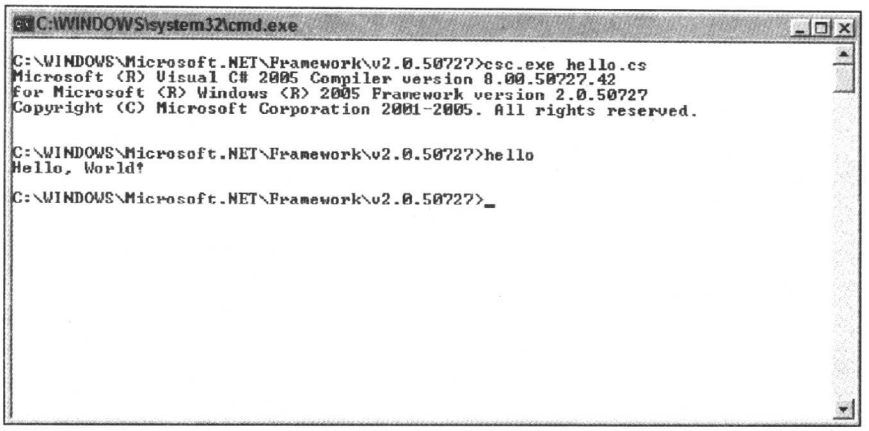


图 1-1

哪怕您对 C#语言知之甚少，也可能已经看出了如何改变显示在屏幕上的信息，这非常简单。例如：

```
public class MyClass
{
  public static void Main()
  {
     System.Console.WriteLine("C# Rules! ");
  }
}
```

代码中这个小小的改变，将屏幕上输出的信息变为“C# Rules!”(如图 1-2 所示)。

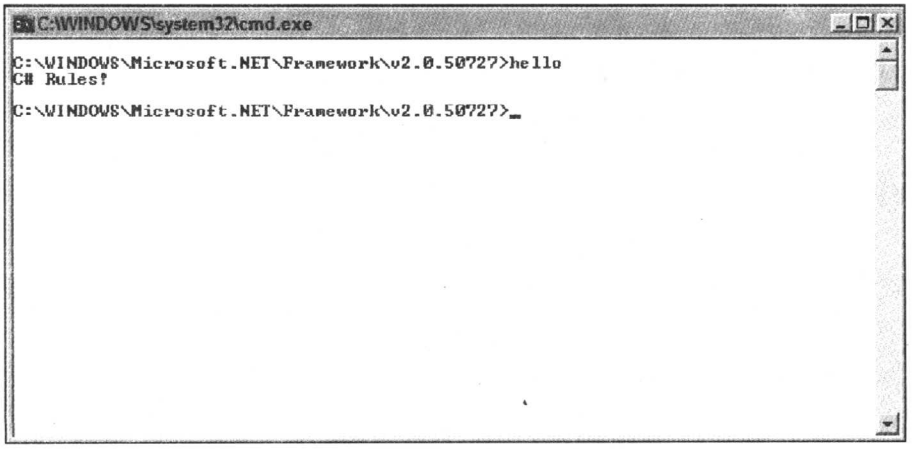


图 1-2