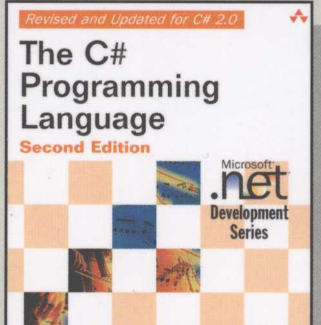
PEARSON
Addison Wesley

典藏
原版书苑

The C# Programming Language
Second Edition

[美] Anders Hejlsberg Scott Wiltamuth Peter Golde 著

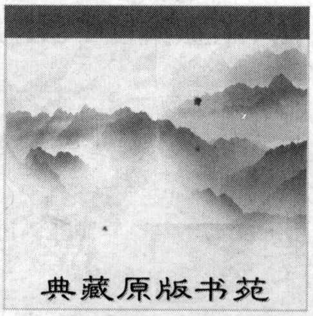
C#编程语言
(第2版)(英文版)



Revised and Updated for C# 2.0
The C# Programming Language
Second Edition
Microsoft .net Development Series
Anders Hejlsberg
Scott Wiltamuth
Peter Golde

C#语言创始人权威著作

人民邮电出版社
POSTS & TELECOM PRESS



C#编程语言

（第2版）（英文版）

Anders Hejlsberg
[美] Scott Wiltamuth 著
Peter Golde

人民邮电出版社
北 京

图书在版编目（CIP）数据

C#编程语言：第 2 版. 英文 /（美）海吉斯博格（Hejlsberg, A.）等著.
—北京：人民邮电出版社，2007.8
（典藏原版书苑）

ISBN 978-7-115-16228-1

Ⅰ. C… Ⅱ. 海… Ⅲ. C 语言—程序设计—英文 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2007）第 067561 号

典藏原版书苑

C#编程语言（第 2 版）（英文版）

◆ 著 ［美］Anders Hejlsberg Scott Wiltamuth Peter Golde
责任编辑 付 飞

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京铭成印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：800×1000 1/16
印张：44.75
字数：963 千字 2007 年 8 月第 1 版
印数：1 – 3 000 册 2007 年 8 月北京第 1 次印刷

著作权合同登记号 图字：01-2007-1479 号

ISBN 978-7-115-16228-1/TP

定价：95.00 元

读者服务热线：（010）67132705 印装质量热线：（010）67129223

内容提要

本书以通俗易懂的语言、精辟丰富的实例，从对 C#的简介开始，全面讲解了 C# 编程语言规范以及各个层面的特性，并且提供了 C# 设计小组的代码示例。本书第 2 版针对 C# 2.0 进行了全面升级，是 C# 权威参考书籍。本书第一部分以 C#语言概述开篇，阐明 C# 语言的概念，接下来对随 Visual Studio .NET 2002 和 2003 发布的 C# 1.0 做了细致完整的技术讲解，内容包括 C#的词法结构、类型、变量、表达式、语句、名字空间、例外、属性和不安全代码等主题。第二部分描述 C# 2.0 的众多特性，包括 Generics、匿名方法（Anonymous Methods）、迭代子（Iterators）、不完整类型（Partial Type）和空类型（Nullable Type）。第 2 版在第 1 版的基础上作了大量的增补和改进，并附有参考资料列表和详尽的索引，能使读者更有效地检索正文并快速找到最感兴趣的主题。

本书作者均为微软 C#开发团队的核心成员，第一作者更是被誉为编程界的神话，“跟 Anders 学 C#”已成全世界读者的普遍共识。本书向读者提供权威、详尽的指导，是 C# 程序员必备的参考书。

前　言

C#项目是从 7 年多以前，也就是 1998 年的 12 月开始的，其目标是创造一种简单、现代、面向对象并且类型安全的新编程语言，也就是现在的.NET 平台。从那时起，C#经历了漫长的历程，现在已经有数十万程序员在使用它，ECMA 和 ISO/IEC 都为它建立了标准，第 2 版包括几种主要的新特性，其开发工作已经完成。

这是一本 C#编程语言的详尽参考书。全书共分三部分：第一部分“C#1.0”，包括第 1 章~第 18 章，对随 Visual Studio .NET 2002/2003 发布的 C# 1.0 做了细致完整的技术讲解；第二部分“C#2.0”，包括第 19 章~第 25 章，描述 C# 2.0 的新增特性，包括 Generics、匿名方法（Anonymous Methods）、迭代子（Iterators）、不完整类型（Partial Type）和空类型（Nullable Type）等；第三部分“附录”，描述了文档注释（documentation comments），总结了 C#2.0 的词法和语法（lexical and syntactic grammars）等内容。

很多人参与了 C#语言的创造。C# 1.0 的设计团队成员有 Anders Hejlsberg、Peter Golde、Peter Hallam、Shon Katzenberger、Todd Proebsting 和 Anson Horton。此外，C# 中 generic 的设计与执行和.NET 通用语言运行时（Common Language Runtime）基于微软研究院的 Don Syme 和 Andrew Kennedy 建立的“Gyro”原型。本书的第 2 版最终由 Mads Torgersen 编辑而成。

在这里无法向所有为 C#设计做出贡献的人们表达谢意。虽然如此，我们还是要在这里感谢他们。没有一个好的设计能够凭空产生，我们从广大热情的开发人员那里得到了持续不断的反馈，这些信息是无价的。

C#一直并且还将是我们参与的最具有挑战性、最激动人心的项目之一。我们希望您享受使用 C#的过程，就像我们创造它时一样。

Anders Hejlsberg

Scott Wiltamuth

2006 年 5 月于西雅图

Contents

PART II C# 2.0 455

Part I

C# 1.0

1. Introduction

C# (pronounced "See Sharp") is a simple, modern, object-oriented, and type-safe programming language. C# has its roots in the C family of languages and will be immediately familiar to C, C++, and Java programmers. C# is standardized by ECMA International as the ***ECMA-334*** standard and by ISO/IEC as the ***ISO/IEC 23270*** standard. Microsoft's C# compiler for the .NET Framework is a conforming implementation of both of these standards.

C# is an object-oriented language, but C# further includes support for ***component-oriented*** programming. Contemporary software design increasingly relies on software components in the form of self-contained and self-describing packages of functionality. Key to such components is that they present a programming model with properties, methods, and events; they have attributes that provide declarative information about the component; and they incorporate their own documentation. C# provides language constructs to directly support these concepts, making C# a very natural language in which to create and use software components.

Several C# features aid in the construction of robust and durable applications: ***Garbage collection*** automatically reclaims memory occupied by unused objects; ***exception handling*** provides a structured and extensible approach to error detection and recovery; and the ***type-safe*** design of the language makes it impossible to have uninitialized variables, to index arrays beyond their bounds, or to perform unchecked type casts.

C# has a ***unified type system***. All C# types, including primitive types such as `int` and `double`, inherit from a single root `object` type. Thus, all types share a set of common operations, and values of any type can be stored, transported, and operated upon in a consistent manner. Furthermore, C# supports both user-defined reference types and value types, allowing dynamic allocation of objects as well as in-line storage of lightweight structures.

To ensure that C# programs and libraries can evolve over time in a compatible manner, much emphasis has been placed on ***versioning*** in C#'s design. Many programming languages pay little attention to this issue, and, as a result, programs written in those languages break more often than necessary when newer versions of dependent libraries are introduced. Aspects of C#'s design that were directly influenced by versioning considerations include the separate `virtual` and `override` modifiers, the rules for method overload resolution, and support for explicit interface member declarations.

The rest of this chapter describes the essential features of the C# language. Although later chapters describe rules and exceptions in a detail-oriented and sometimes mathematical manner, this chapter strives for clarity and brevity at the expense of completeness. The intent is to provide the reader with an introduction to the language that will facilitate the writing of early programs and the reading of later chapters.

1.1 Hello World

The "Hello, World" program is traditionally used to introduce a programming language. Here it is in C#:

```
using System;

class Hello
{
    static void Main() {
        Console.WriteLine("Hello, World");
    }
}
```

C# source files typically have the file extension `.cs`. Assuming that the "Hello, World" program is stored in the file `hello.cs`, the program can be compiled with the Microsoft C# compiler using the command line

```
csc hello.cs
```

which produces an executable assembly named `hello.exe`. The output produced by this application when it is run is

```
Hello, World
```

The "Hello, World" program starts with a `using` directive that references the `System` namespace. Namespaces provide a hierarchical means of organizing C# programs and libraries. Namespaces contain types and other namespaces—for example, the `System` namespace contains a number of types, such as the `Console` class referenced in the program, and a number of other namespaces, such as `IO` and `Collections`. A `using` directive that references a given namespace enables unqualified use of the types that are members of that namespace. Because of the `using` directive, the program can use `Console.WriteLine` as shorthand for `System.Console.WriteLine`.

The `Hello` class declared by the "Hello, World" program has a single member, the method named `Main`. The `Main` method is declared with the `static` modifier. Unlike instance methods, which reference a particular object instance using the keyword `this`, static methods operate without reference to a particular object. By convention, a static method named `Main` serves as the entry point of a program.