开发人员专业技术丛书

HZ BOOKS

The C# Programming Language

Third Edition

C#程序设计语言

（原书第3版）

（美） Anders Hejlsberg
Mads Torgersen
Scott Wiltamuth
Peter Golde 著

顾雁宏 徐旭铭 译

机械工业出版社
China Machine Press

开发人员专业技术丛书

The C# Programming Language

Third Edition

C#程序设计语言

（原书第3版）

（美）Anders Hejlsberg
Mads Torgersen
Scott Wiltamuth
Peter Golde 著

顾雁宏 徐旭铭 译

机械工业出版社
China Machine Press

C#语言结合了快速应用开发语言的高效和C/C++语言的强大。现在C# 3.0又加入了函数式编程技术和语言集成查询（Language INtegrated Query，LINQ）。本书正是C# 3.0的权威技术指南。

这一版由C#的缔造者Anders Hejlsberg和他的同事Mads Torgersen、Scott Wiltamuth和Peter Golde编写，全部内容都更新到了C# 3.0版。本书提供了C#语言完整的规格说明，以及大量的描述、参考资料、范例代码，和来自9位卓越的C#大师的详细注解。

这些注解（这一版新的特色）所达到的深度和广度是很难在其他书中找到的。本书的正文介绍了C#的概念，而这些恰到好处的注解则解释了为什么这些特性是重要的，应该怎么使用它们，它们和其他语言的关系是什么，以及它们是如何进化而来的。

对任何希望深入理解C#的程序员来说，本书都是绝对不容错过的参考经典。

本书版权登记号：图字：01-2009-4560

图书在版编目（CIP）数据

C#程序设计语言（原书第3版）/（美）海杰尔斯伯格（Hejlsberg, A.）等著；顾雁宏等译. —北京：机械工业出版社，2009.9

（开发人员专业技术丛书）

书名原文：C# Programming Language, Third Edition

ISBN 978-7-111-28261-7

Ⅰ.C…　Ⅱ.①海…　②顾…　Ⅲ.C语言—程序设计　Ⅳ.TP312

中国版本图书馆CIP数据核字（2009）第161240号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：陈佳媛

北京京师印务有限公司印刷

2010年1月第1版第1次印刷

186mm × 240mm · 35.25印张

标准书号：ISBN 978-7-111-28261-7

定价：79.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991，88361066

购书热线：（010）68326294，88379649，68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

序

自2000年夏.NET发布以来已经有8个年头了。对我来说，当时.NET最重要的两点就是结合了托管代码的本地执行和用于程序之间通信的XML消息机制。不过那个时候我还没有意识到C#会变得那么重要。

C#从一开始就是程序员们理解和使用.NET的主要手段。如果你问一个普通的.NET程序员一个值类型和一个引用类型的区别是什么，通常的回答都是“结构和类的区别”，而非“是否是从System.ValueType继承而来的类型”。为什么？因为我们都是用语言，而不是通过API来和运行时（更重要的是，其他人）交流想法和意图的。

如果没有一门出色的语言，一个平台要想成功是不可想象的。C#最初就为人们如何看待.NET打下了坚实的基础。随着.NET的不断发展，C#的重要性也与日俱增，诸如迭代器和真正的闭包（也叫匿名方法）都是用C#编译器实现的纯语言特性，而不是平台特性。C# 3.0的发布更意味着C#成为了.NET不断创新的主角，它引入了标准化的查询操作符，简洁的lambda表达式、扩展方法，以及在运行时访问表达式树的能力——这些都是通过语言和编译器实现的。

说到C#就一定会提到它的缔造者Anders Hejlsberg。我非常荣幸地在C# 3.0设计阶段连续好几个月参与了C#的设计会议，Anders的工作让我大开眼界。他那种深谙程序员喜欢什么和不喜欢什么的天赋实在是一流——同时他又能和设计团队紧密合作，并最终获得最佳的设计方案。

特别是在C# 3.0上，Anders在从函数式语言社区获取灵感并将它们带给广大群众的过程中展现出无与伦比的能力。要知道这绝对不是一件容易的事情。Guy Steele曾经在谈论Java时说到：“我们没打算要吸引Lisp程序员，我们的目标是C++程序员。我们成功地把他们从转向Lisp的路上吸引了过来。”当我看到C# 3.0的时候，我就知道C#已经至少获得了一名C++程序员（就是我自己）的青睐。

即使C#很出色，但是人们还是需要一份用自然语言（这里是英文）和一些范式（BNF）写成的文档来帮助他们抓住要点，以及任何消除晦涩的地方。而你手中的这本书正是这样的一份文档。据我的经验，我敢说每个.NET程序员在读这本书的时候都至少会有一次“啊，原来如此”的感叹，它能让你的水平更上一层楼。

享受它吧。

Don Box

2008年7月

作 者 简 介

Anders Hejlsberg是编程界的传奇人物。他是C#语言的架构师，同时也是微软技术专家。他在1996年加入微软，之前13年的职业生涯则是在Borland度过，他曾经是Delphi和Turbo Pascal的首席架构师。

Mads Torgersen是微软的资深程序经理。作为C#的程序经理，他负责召开C#语言的设计会议及维护C#语言的规范。在2005年加入微软之前，Mads是奥尔胡斯大学的副教授，主要教授和研究面向对象编程语言。在那里，他领导的小组设计实现了Java的泛型通配符。

Scott Wiltamuth是Visual Studio的合作程序经理。他在微软做过很多面向程序员的项目，包括Visual Basic、VBScript、Jscript、Visual J++和Visual C#。Scott是C#语言的设计师之一，他拥有斯坦福大学计算机科学硕士学位。

Peter Golde在离开微软之前是微软C#编译器的首席程序员。他作为微软在ECMA委员会（这个委员会负责了C#的标准化工作）的主要代表，领导实现了编译器并参与了语言的设计。

注解者简介

Brad Abrams是CLR和.NET架构团队在微软的创始人之一。他从1998年起就参与了.NET架构的设计，现在他是微软UI框架和服务团队的首席程序经理。在Brad的博客http://blogs.msdn.com/BradA上可以读到他最新的想法。

Joseph Albahari是《C# 3.0 in a Nutshell》（O'Reilly，2007）、《C# 3.0 Pocket Reference》（O'Reilly，2008）和《LINQ Pocket Reference》（O'Reilly，2008）的合著者。他在保健、教育和电信领域里作为资深程序员和软件架构师有17年的经验，他还是LINQPad的作者，这是一个可以用LINQ交互查询数据库的工具。

Krzysztof Cwalina是.NET架构团队的程序经理。他最初在微软负责了Framework 1.0的API设计。现在他主要负责开发、引导和推行.NET架构的设计思想。他是《Framework Design Guidelines》（Addison-Wesley，2005）的作者之一。他的博客是http://blogs.msdn.com/kcwalina。

Don Box是微软杰出的工程师，他开发的声明式语言和工具大大简化了应用程序和服务的开发，同时他还参与设计了语言、框架和第一手的用户体验来帮助人们把他们对软件的想法和要求变成可执行的代码。Don在软件开发领域还是一名广受尊敬的作家，负责编辑了微软.NET开发系列丛书（Addison-Wesley），他还参与编辑了《C++ Report》，《Microsoft Systems Journal（MSJ）》和《MSDN Magazine》。

Jesse Liberty（"Silverlight专家"）是微软的高级程序经理，还是好几本畅销书的作者，包括了O'Reilly Media即将出版的《Programming Silverlight 2》和《Programming C# 3.0》。Jesse有20多年的编程经历。他的个人主页是www.JesseLiberty.com。

Eric Lippert是微软C#编译器团队的资深程序员。他参与设计和实现了Visual Basic、VBScript、Jscript、C#和Visual Studio Tools for Office。他的博客上全是这方面的资料，网址是http://blogs.msdn.com/EricLippert。

Fritz Onion是.NET程序员培训公司Pluralsight的创办人之一。Fritz领衔了Pluralsight的Web开发课程，在全球教授ASP.NET、Ajax和Silverlight。他是广受好评的《Essential ASP.NET》和《Essential ASP.NET 2.0》（都是由Addison-Wesley出版）的作者。同时他还为很多专业的程序员杂志撰写文章。他的博客是http://pluralsight.com/fritz。

Vladimir Reshetnikov是微软Visual C#的MVP。他有超过6年的软件开发经验，以及4年Microsoft.NET和C#的经验。

Chris Sells是微软Connected Systems分部的程序经理。Chris编写了好几本书，包括了《Programming WPF》（O'Reilly，2007）、《Windows Forms 2.0 Programming》（Addison-Wesley，2006）和《ATL Internal》（Addison-Wesley，1999）。闲暇的时候，他还主持了很多研讨会，并且在微软内部的产品讨论组里也相当活跃。要想了解Chris和他的项目，请访问www.sellsbrothers.com。

Bill Wagner是SRT Solutions的创始人、微软的区域主管和一名C# MVP。他是《Effective C#》（Addison-Wesley，2005）和《More Effective C#》（Addison-Wesley，2009）的作者，《Visual Studio Magazine》的专栏作家，同时还是一位C#开发者中心的贡献者。

前　言

C#项目始于10年之前的1998年12月，当初的目标是要为全新的（还未命名的）.NET平台创造一种简单、现代化、面对对象和类型安全的编程语言。一路走来，C#也算是历经坎坷。现在这门语言已经有超过一百万的程序员，发布了3个版本，其中每一版都加入许多重大的新特性。

而本书也一样出到了第3版。作为一本C#语言完整的技术规范，第3版和之前的两个版本有很多不同的地方。其中最显著的当然就是它覆盖了所有C# 3.0的新特性，包括对象和集合初始化、匿名类型、lambda表达式、查询表达式和局部方法。绝大多数这些特性都是为了支持一种更加函数式和声明式的编程风格，具体来讲，就是语言集成查询（LINQ），它提供了一种统一的方式来查询不同数据源的数据。而LINQ又大量构建在一些C# 2.0引入的特性之上，比如泛型、迭代器和局部类型。

第3版中的另一个变化是所有的规范说明都重新组织了。在第2版里，C# 2.0引入的特性和原本C# 1.0中的特性是分开描述的。要是再加上一套新特性的话，这个方法就没用了——读者需要从3个不同的地方获取相关的信息，本书的实用性将会大打折扣。所以，本书会按照主题来组织内容，3个语言版本中的特性被放在一起来介绍。

最后一个和之前版本区别的地方就是本书包含了大量的注解。我们很幸运地请到了一些世界级的C#和.NET专家，在书中各处以注解的形式为我们提供了出色的指导、背景和观点。我们很高兴这些注解和本书的核心内容相辅相成，让C#的这些特性跃然纸上。

创造C#语言是很多人共同努力的结果。C# 1.0的设计团队由Anders Hejlsberg、Scott Wiltamuth、Peter Golde、Peter Sollich和Eric Gunnerson组成。而C# 2.0团队的成员有Anders Hejlsberg、Peter Golde、Peter Hallam、Shon Katzenberger、Todd Proebsting和Anson Horton。此外，C#和.NET公共语言运行时（Common Language Runtime）中泛型的设计和实现是基于微软研究院里Don Syme和Andrew Kennedy构建的“Gyro”原型之上。C# 3.0则是由Anders Hejlsberg、Peter Hallam、Shon Katzenberger、Dinesh Kulkarni、Erik Meijer、Mads Torgersen和Matt Warren负责设计。

要感谢所有影响了C#设计的人是不可能的，尽管如此我们还是要感谢你们。闭门造车是不会有好设计的，所以来自我们庞大和热情的程序员社区的意见和建议都是无价的。

C#是（并将继续是）我们工作过的最有挑战性和最刺激的项目之一。希望你们用得开心，我们也做得高兴。

Anders Hejlsberg
Mads Torgersen
Scott Wiltamuth
华盛顿西雅图
2008年7月

目　录

第1章 介 绍

C#（发音同“see sharp”）是一门简单、现代化、面对对象和类型安全的编程语言。C#属于C语言家族，任何C、C++或Java程序员都不会觉得它很陌生。C#在ECMA International的标准是ECMA-334，在ISO/IEC上的标准是ISO/IEC 23270。微软.NET Framework上的C#编译器的实现同时遵循了这两个标准。

C#不只是一门面对对象的语言，它还包含了对面向组件（component-oriented）编程的支持。现代的软件设计越来越依赖于自包含和自描述的功能包形式的软件组件。这类组件的重要之处在于它们提供了一个带有属性（property）、方法（method）和事件（event）的编程模型，拥有提供了关于组件声明信息的特性（attribute），另外它们还包含了自身的文档。C#在语言级别上直接支持了这些概念，令C#可以很自然地创建和使用软件组件。

C#还提供了一些特性来帮助构建健壮、耐用的应用程序：垃圾收集（Garbage Collection）会自动回收不再使用的对象所占用的内存；异常处理（exception hardling）提供了一种结构化且可扩展的方式来检测错误和恢复；而语言的类型安全（type-safe）设计则可以防止读取未初始化的变量、数组越界或进行未检查的类型转换。

C#拥有统一的类型系统（unified type system）。所有的C#类型，包括int和double这样的基础类型，都是从根类型object继承而来。所以，所有的类型都具有一些通用的操作，任何类型的值都可以通过一致的方式进行保存、传递和操作。此外，C#还支持用户自定义引用类型和值类型，允许动态分配对象和轻型结构的内联（in-line）存储。

为了保证C#程序和类库能以兼容的方式向前发展，C#在设计过程中非常注意版本控制（versioning）。很多编程语言都对这一点重视不够，所以当新版本的依赖库被引入时，用这些语言编写的程序都无谓地失灵了。C#设计中的很多方面都直接受到了版本控制考量的影响，包括区分virtual和override修饰符、方法重载的规则，以及对接口成员显式声明的支持。

在本章的剩余部分里，我们将介绍C#语言的重要特性。稍后的章节会更细致（更精确）地讲解一些规则和例外，不过这里先尽量做一个简明的介绍。这让读者可以先对语言有一个初步的认识，以便及早开始编程和理解稍后的章节。

> CHRIS SELLS 我绝对同意“现代化、面对对象和类型安全”的论述，不过C#已经不再是一门简单的语言了。但是，随着C# 2.0加入的泛型和匿名委托，以及C# 3.0引入的LINQ特性，其程序本身正在变得更加简单易读，维护起来也更加容易——这些正是任何编程语言的梦想。

> ERIC LIPPERT C#还越来越像一门函数式编程语言。诸如类型推导（type inference）、lambda表达式和一元查询推导式（monadic query comprehension）这样的特性让传统的面对对象程序员可以利用函数式编程的思想来增加语言的表达能力。

1.1 Hello, World

介绍编程语言时最经典的自然就是“Hello, World”程序了。这里是C#版：

```
using System;

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

C#的源文件通常是以.cs结尾。假设这个“Hello, World”程序被保存为hello.cs，那么就可以在命令行下用微软C#编译器这样编译程序

```
csc hello.cs
```

它会生成一个可执行文件hello.exe。这个程序执行的输出为

```
Hello, World
```

“Hello, World”程序的开始通过using指令引用了System命名空间。命名空间提供了一种层次化的方式来组织C#程序和类库。它可以包含类型和其他命名空间，例如，System命名空间包含了诸如在程序中引用的Console类的多种类型；以及其他一些命名空间，比如IO和Collections。通过using指令引用某个命名空间时，你就可以不加前缀地使用这个命名空间里的类型。在这个程序里的using指令让程序可以把System.Console.WriteLine简写为Console.WriteLine。

“Hello, World”程序声明的Hello类只有一个成员方法Main。Main方法在声明时要带上static修饰符。实例方法可以通过关键字this来引用自身，而静态方法在使用时却无需引用某个特定对象。按照惯例，静态方法Main是程序的入口点。

程序的输出是由命名空间System里的Console类的WriteLine方法产生的。在默认情况下，微软的C#编译器会自动引用这个由.NET框架类库提供的类。注意C#本身并没有一个单独的运行库，.NET框架就是C#的运行库。

BRAD ABRAMS 请注意：有趣的是这里Console.WriteLine()其实是Console.Out.WriteLine的简写。Console.Out是一个属性，它返回了一个设计用来专门向控制台输出的System.IO.TextWriter基类的实现。上面的例子写成这样也是正确的：

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

在框架设计的初期，我们就注意到了在C#语言规格说明中的这节给语言所带来的复杂性。我们选择了方便地重载Console来让“Hello, World”更加容易编写。事实上，今天你会发现你几乎用不着去调用Console.Out.WriteLine()了，大家也都很喜欢这个决定。

1.2 程序结构

在C#中关键的组织概念是*程序*、*命名空间*、*类型*、*成员*和*汇编代码*。C#的程序由一个或多个源文件组成。程序声明了类型，而类型包含了成员。类型可以被组织到命名空间里。类和接口都是类型的例子。变量、方法、属性和事件则是成员的例子。当编译C#程序的时候，它们通常都会被打包到汇编代码里去。它们的文件扩展名可是.exe或是.dll，分别取决于它们实现的是*应用程序*还是*类库*。

下面这个例子

```
using System;

namespace Acme.Collections
{
    public class Stack
    {
        Entry top;

        public void Push(object data) {
            top = new Entry(top, data);
        }

        public object Pop() {
            if (top == null) throw new InvalidOperationException();
            object result = top.data;
            top = top.next;
            return result;
        }

        class Entry
        {
            public Entry next;
            public object data;

            public Entry(Entry next, object data) {
                this.next = next;
                this.data = data;
            }
        }
    }
}
```

在命名空间Acme.Collections里声明了一个Stack类，所以这个类完整的名字就是Acme.Collections.Stack。它包含了几个成员：一个变量top，两个方法Push和Pop，还有一个嵌套类Entry。Entry类又包含了3个成员：一个next变量，一个data变量，一个构造函数。假设这个例子的源代码被保存在文件acme.cs里，那么命令

```
csc /t:library acme.cs
```

会把它编译成一个类库（没有Main入口点的代码）并生成一个汇编文件acme.dll。

汇编文件包含了*中间语言*（Intermediate Language，IL）指令形式的可执行代码，以及*元数据*（metadata）形式的链接信息。在执行之前，IL代码会被.NET 公共语言运行库的即时编译