

Illustrated C# 2010

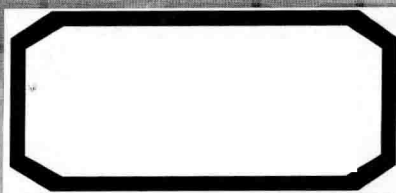
C# 4.0图解教程

[美] Daniel M. Solis 著
苏林 朱晔 等译

- 用图说话，最易学的C#教程
- Amazon全五星盛誉
- 涵盖C# 4.0和.NET 4最新特征

TURING

图灵程序设计丛书



Illustrated C# 2010

C# 4.0图解教程

[美] Daniel M. Solis 著
苏林 朱晔 等译



人民邮电出版社
北京

图书在版编目 (CIP) 数据

C# 4.0图解教程 / (美) 索利斯 (Solis, D.M.) 著 ; 苏林等译. — 北京 : 人民邮电出版社, 2011.6

(图灵程序设计丛书)

书名原文: Illustrated C# 2010

ISBN 978-7-115-25183-1

I. ①C… II. ①索… ②苏… III. ①C语言—程序设计—教材 IV. ①TP312

中国版本图书馆CIP数据核字(2011)第056110号

内 容 提 要

本书是一本广受称赞的 C# 教程。它以图文并茂的形式,用朴实简洁的文字,并辅之以大量表格和代码示例,全面地阐述了最新版 C# 语言的各种特性,使读者能够快速理解、学习和使用 C#。同时,本书还讲解了 C# 与 VB、C++ 等主流语言的不同点和相似之处。

本书是经典的 C# 入门书,不仅适合没有任何编程语言基础的初级读者阅读,而且还是 VB、C++ 等语言基础的 C# 初学者的最佳选择。

图灵程序设计丛书 C# 4.0图解教程

-
- ◆ 著 [美] Daniel M. Solis
 - 译 苏 林 朱 晔 等
 - 责任编辑 卢秀丽
 - ◆ 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 电子邮件 315@ptpress.com.cn
 - 网址 <http://www.ptpress.com.cn>
 - 北京艺辉印刷有限公司印刷
 - ◆ 开本: 800×1000 1/16
 - 印张: 30.25
 - 字数: 715千字 2011年6月第1版
 - 印数: 1-4 000册 2011年6月北京第1次印刷

著作权合同登记号 图字: 01-2011-1373号

ISBN 978-7-115-25183-1

定价: 69.00元

读者服务热线: (010)51095186转604 印装质量热线: (010)67129223

反盗版热线: (010)67171154

版 权 声 明

Original English language edition, entitled *Illustrated C# 2010* by Daniel M. Solis published by Apress, 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705.

Copyright © 2010 by Daniel M. Solis. Simplified Chinese-language edition copyright © 2011 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由Apress L. P.授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

译者序

C#是一门基于.NET的高级语言，正是因为C#处于.NET温暖的怀抱，所以许多C#程序员，甚至许多C#高级程序员对.NET在内存和指令等本质问题上的认识不够。况且有许多使用C#的程序员在使用ASP.NET技术进行网站开发，他们有的从脚本语言转型而来，有的在没有充分学习C#的情况下就投入了开发工作，那么他们可能对本质问题的认识就更差一点。但是笔者认为，不管怎么样，都非常有必要更深入理解语言背后的机制，而不仅仅停留在掌握API使用的层次上。只有这样，你才能意识到很多bug的关键点和性能问题的关键点，并且理解那些高级的特性。

从目录上来看本书就像其他C#入门书籍一样，介绍了一个又一个语言特性，但是如果你翻阅一下正文就会发现它的不同。可能因为作者有C/C++的背景的关系，对于每一个语言特性，作者对其使用方式只是轻描淡写，而对特性背后的机制做了浓墨重彩的介绍，并且在文字介绍中穿插大量图示来展现内存对象的面貌。其实，市面上很多所谓的进阶书籍都只是介绍如何使用那些高级API、高级特性，而忽略了语言本质，但这一块恰巧是最重要的。因此，对于那些用了几年C#的程序员来说，本书具有非常大的价值。

不管怎样，一句话，本书值得一读。但是由于时间仓促，本书在翻译过程中难免出现失误，如果有任何问题，欢迎来信交流，笔者邮箱为yzhu@live.com。

朱 晔

2011年3月

上一版译者序

书是知识的载体，是智慧的传播者。技术图书在技术的普及、发展过程中的作用是毋庸置疑的。在这个知识爆炸、信息技术迅猛发展的时代，技术图书的作用更加突出。我们比以往任何时候都需要关注新技术和新平台的参考资料。一本描述清晰、内容详实的书能使我们快速掌握这些技术。

笔者不才，自己无力写出这样的书，愿意以虫蚁之能，行搬运之事，将优秀外文书籍译成中文，以利于国人参考和学习，从而为技术传播尽自己的绵薄之力。

C#和.NET平台近年来迅速普及，已经成为很多公司使用的主要技术之一。有很多出色的应用都是使用C#开发的，包括很多Web 2.0时代的网络应用。虽然.NET平台目前还只能在Windows操作系统下工作，但是这并没有妨碍它发展壮大。一方面是因为Windows操作系统的普及程度已经给.NET提供了巨大的发展空间；另一方面是因为.NET确实是个优秀的平台，而且C#也确实算得上是新一代的优秀的面向对象编程语言。作为一个与时俱进的软件工程师，忽视C#和.NET是很不明智的。

本书是一部极为出色的C#著作。正如本书作者所说，它不仅包含了入门的基础知识，而且同时还能作为开发过程中的参考书使用。书中使用了大量的示例和图表，使内容一目了然。即便是有经验的C#程序员，阅读这本书也会受益匪浅。

在本书的翻译过程中，我尽量保持原书清晰明了的风格，并努力保证术语及用词的准确。由于能力有限，我虽已尽所能，但仍难免有不妥之处，望读者朋友海涵。

感谢我的妻子毛毛！在我翻译本书的过程中，她承担了大部分的家务，并给予了我很多支持和鼓励。没有她的爱和付出，本书的翻译工作肯定不会进展得如此顺利。

相信这本书一定对你有帮助！

苏 林

2008年5月于上海

谨将此书献给Sian、我的父母Sal和Amy，还有Sue。

前 言

本书的目的是讲授C#编程语言的基础知识和工作原理。大多数编程图书以文字为主要载体。对于小说而言，文字形式当然是最恰当不过了，但对于编程语言中的很多重要概念，综合运用文字、图形和表格会更容易理解。

许多人都习惯于形象思维，而图形和表格有助于我们更清晰地理解概念。在几年的编程语言教学工作中我发现，我在白板上画的图能帮助学生最快理解我要传达的概念。然而，单靠图表并不足以解释一种编程语言和平台。本书的目标是以最佳方式结合文字和图表，使你对这种语言有透彻的理解，并且让本书能用作参考工具。

本书写给所有想要学习C#的人——从初学者到有经验的程序员。刚开始学编程的人会发现，书中全面讲述了基础知识；有经验的程序员会觉得，内容的叙述非常简洁明晰，无需费力卒读就能直接获得想要的信息。无论哪种程序员，内容本身的图形化呈现方式都能帮助你更容易地学习本书。

你可以从Apress网站（apress.com）下载本书所有示例程序的源代码^①。尽管我不能回答有关代码的一些细节问题，但是你可以通过dansolis@sbcglobal.net和我取得联系，提出建议或反馈，还可以访问我的站点illustratedcsharp.com。最后，如果你有兴趣对学习使用Windows Presentation Foundation来编程，可以阅读我的另外一本书——*Illustrated WPF*，该书的风格及叙述方式和本书一样。

我希望本书可以让你享受学习C#的过程！祝你好运！

致谢

我想感谢Sian每天支持并鼓励我，我还想感谢我的父母、兄弟和姐妹，他们一直爱我并支持我。

我还想对Apress的朋友表达诚挚的感谢，是他们与我携手完成了本书。我真心感激他们理解并赞赏我努力做的事情，并和我一起完成它。感谢你们所有人。

^① 本书源代码也可以从图灵网站www.turingbook.com本书网页免费注册下载。——编者注

目 录

第 1 章 C#和.NET 框架 1	
1.1 在.NET 之前..... 1	
1.1.1 20 世纪 90 年代后期的 Windows 编程..... 1	
1.1.2 下一代平台服务的目标..... 2	
1.2 进入 Microsoft .NET..... 2	
1.2.1 .NET 框架的组成..... 2	
1.2.2 大大改进的编程环境..... 3	
1.3 编译成 CIL..... 5	
1.4 编译成本机代码并执行..... 5	
1.5 CLR..... 6	
1.6 CLI..... 7	
1.7 缩写回顾..... 8	
第 2 章 C#编程概述 9	
2.1 一个简单的 C#程序..... 9	
2.2 标识符和关键字..... 11	
2.2.1 命名约定..... 11	
2.2.2 关键字..... 12	
2.3 Main: 程序的起始点..... 12	
2.4 空白..... 13	
2.5 语句..... 13	
2.5.1 简单语句..... 13	
2.5.2 块..... 14	
2.6 从程序中输出文本..... 14	
2.6.1 Write..... 14	
2.6.2 WriteLine..... 15	
2.6.3 格式字符串..... 15	
2.6.4 多重标记和值..... 16	
2.7 注释..... 16	
2.7.1 关于注释的补充..... 17	
2.7.2 文档注释..... 17	
2.7.3 注释类型总结..... 18	
第 3 章 类型、存储和变量 19	
3.1 C#程序是一组类型声明..... 19	
3.2 类型是一种模板..... 20	
3.3 实例化类型..... 20	
3.4 数据成员和函数成员..... 21	
3.5 预定义类型..... 21	
3.6 用户定义类型..... 23	
3.7 栈和堆..... 24	
3.7.1 栈..... 24	
3.7.2 堆..... 25	
3.8 值类型和引用类型..... 25	
3.8.1 存储引用类型对象的成员..... 26	
3.8.2 C#类型的分类..... 27	
3.9 变量..... 27	
3.9.1 变量声明..... 27	
3.9.2 多重变量声明..... 29	
3.9.3 使用变量的值..... 29	
3.10 静态类型和 dynamic 关键字..... 29	
3.11 可空类型..... 30	
3.11.1 创建可空类型..... 30	
3.11.2 为可空类型赋值..... 31	
第 4 章 类：基础 32	
4.1 类的概述..... 32	
4.2 程序和类：一个快速的示例..... 33	
4.3 声明类..... 33	

4.4	类成员	34	6.2	实例类成员	76
4.4.1	字段	34	6.3	静态字段	76
4.4.2	显式和隐式字段初始化	35	6.4	从类的外部访问静态成员	77
4.4.3	声明多个字段	36	6.4.1	静态字段示例	77
4.4.4	方法	36	6.4.2	静态成员的生存期	78
4.5	创建变量和类的实例	37	6.5	静态函数成员	79
4.6	为数据分配内存	37	6.6	其他静态类成员类型	80
4.7	实例成员	38	6.7	成员常量	80
4.8	访问修饰符	39	6.8	属性	82
4.9	从类的内部访问成员	41	6.8.1	属性声明和访问器	83
4.10	从类的外部访问成员	42	6.8.2	属性示例	83
4.11	综合应用	43	6.8.3	使用属性	84
第 5 章	方法	45	6.8.4	属性和关联字段	85
5.1	方法的结构	45	6.8.5	执行其他计算	86
5.2	本地变量	47	6.8.6	只读和只写属性	87
5.2.1	类型推断和 var 关键字	47	6.8.7	计算只读属性示例	87
5.2.2	嵌套块中的本地变量	48	6.8.8	属性和数据库示例	88
5.3	本地常量	49	6.8.9	属性 vs. 公共字段	88
5.4	方法调用	51	6.8.10	自动实现属性	89
5.5	返回值	52	6.8.11	静态属性	89
5.6	参数	54	6.9	实例构造函数	90
5.6.1	形参	54	6.9.1	带参数的构造函数	91
5.6.2	实参	55	6.9.2	默认构造函数	92
5.6.3	带位置输入参数的方法示例	55	6.10	静态构造函数	93
5.7	值参数	56	6.10.1	静态构造函数示例	93
5.8	引用参数	58	6.10.2	构造函数的可访问性	94
5.9	输出参数	60	6.11	对象初始化列表	94
5.10	参数数组	62	6.12	析构函数	95
5.10.1	方法调用	63	6.12.1	调用析构函数	96
5.10.2	数组作实参	65	6.12.2	标准清理模式	98
5.11	参数类型总结	66	6.13	比较构造函数和析构函数	99
5.12	方法重载	66	6.14	readonly 修饰符	99
5.13	命名参数	67	6.15	this 关键字	100
5.14	可选参数	68	6.16	索引	101
5.15	栈帧	71	6.16.1	什么是索引	102
5.16	递归	72	6.16.2	索引和属性	102
第 6 章	类进阶	74	6.16.3	声明索引	103
6.1	类成员	74	6.16.4	索引的 set 访问器	104
			6.16.5	索引的 get 访问器	104
			6.16.6	关于索引的补充	105

6.16.7	为 Employee 示例声明索引	105	8.4	字符字面量	145
6.16.8	另一个索引示例	106	8.5	字符串字面量	146
6.16.9	索引重载	107	8.6	求值顺序	147
6.17	访问器的访问修饰符	107	8.6.1	优先级	148
6.18	分部类和分部类型	109	8.6.2	结合性	148
第 7 章	类和继承	112	8.7	简单算术运算符	149
7.1	类继承	112	8.8	求余运算符	149
7.2	访问继承的成员	113	8.9	关系比较运算符和相等比较运算符	150
7.3	隐藏基类的成员	115	8.10	递增运算符和递减运算符	152
7.4	基类访问	116	8.11	条件逻辑运算符	153
7.5	使用基类的引用	117	8.12	逻辑运算符	154
7.5.1	虚方法和覆写方法	119	8.13	移位运算符	155
7.5.2	覆写标记为 override 的 方法	120	8.14	赋值运算符	157
7.5.3	覆盖其他成员类型	123	8.15	条件运算符	158
7.6	构造函数的执行	124	8.16	一元算术运算符	159
7.6.1	构造函数初始化语句	125	8.17	用户定义类型转换	160
7.6.2	类访问修饰符	127	8.18	运算符重载	162
7.7	程序集间的继承	128	8.18.1	运算符重载的限制	163
7.8	成员访问修饰符	129	8.18.2	运算符重载的示例	164
7.8.1	访问成员的区域	130	8.19	typeof 运算符	165
7.8.2	公有成员的可访问性	131	8.20	其他运算符	166
7.8.3	私有成员的可访问性	131	第 9 章	语句	167
7.8.4	受保护成员的可访问性	132	9.1	什么是语句	167
7.8.5	内部成员的可访问性	132	9.2	表达式语句	168
7.8.6	受保护内部成员的可访问性	132	9.3	控制流语句	169
7.8.7	成员访问修饰符的总结	133	9.4	if 语句	169
7.9	抽象成员	133	9.5	if...else 语句	170
7.10	抽象类	134	9.6	switch 语句	171
7.10.1	抽象类和抽象方法的示例	135	9.6.1	分支示例	172
7.10.2	抽象类的另一个例子	136	9.6.2	switch 语句的补充	173
7.11	密封类	137	9.6.3	switch 标签	174
7.12	静态类	137	9.7	while 循环	174
7.13	扩展方法	138	9.8	do 循环	175
第 8 章	表达式和运算符	142	9.9	for 循环	176
8.1	表达式	142	9.9.1	for 语句中变量的有效范围	177
8.2	字面量	143	9.9.2	初始化语句和迭代表达式 中的多表达式	178
8.3	整数字面量	144	9.10	跳转语句	178
			9.11	break 语句	178

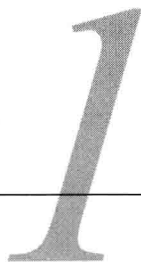
9.12	continue 语句	179	11.5	finally 块	210
9.13	标签语句	180	11.6	为异常寻找处理代码	211
9.13.1	标签	180	11.7	更进一步搜索	211
9.13.2	标签语句的范围	180	11.7.1	一般法则	212
9.14	goto 语句	181	11.7.2	搜索调用栈的示例	213
9.15	using 语句	182	11.8	抛出异常	214
9.15.1	资源的包装使用	183	11.9	不带异常对象的抛出	215
9.15.2	using 语句的示例	183			
9.15.3	多个资源和嵌套	184	第 12 章	结构	217
9.15.4	using 语句的另一种形式	185	12.1	什么是结构	217
9.16	其他语句	186	12.2	结构是值类型	218
第 10 章	命名空间和程序集	187	12.3	对结构赋值	219
10.1	引用其他程序集	187	12.4	构造函数和析构函数	220
10.2	命名空间	191	12.4.1	实例构造函数	220
10.2.1	命名空间名称	194	12.4.2	静态构造函数	221
10.2.2	命名空间的补充	194	12.4.3	构造函数和析构函数的 总结	221
10.2.3	命名空间跨文件伸展	195	12.5	字段初始化是不允许的	222
10.2.4	嵌套命名空间	195	12.6	结构是密封的	222
10.3	using 指令	196	12.7	装箱和取消装箱	222
10.3.1	using 命名空间指令	196	12.8	结构作为返回值和参数	222
10.3.2	using 别名指令	197	12.9	关于结构的附加信息	222
10.4	程序集的结构	198	第 13 章	枚举	224
10.5	程序集标识符	199	13.1	枚举	224
10.6	强命名程序集	200	13.1.1	设置底层类型和显式值	225
10.7	程序集的私有方式部署	201	13.1.2	隐式成员编号	226
10.8	共享程序集和 GAC	201	13.2	位标志	227
10.8.1	把程序集安装到 GAC	201	13.2.1	Flags 特性	229
10.8.2	GAC 内的并肩执行	202	13.2.2	使用位标志的示例	230
10.9	配置文件	203	13.3	关于枚举的补充	231
10.10	延迟签名	203	第 14 章	数组	233
第 11 章	异常	205	14.1	数组	233
11.1	什么是异常	205	14.1.1	定义	233
11.2	try 语句	206	14.1.2	重要细节	234
11.3	异常类	207	14.2	数组的类型	234
11.4	catch 子句	207	14.3	数组是对象	235
11.4.1	使用特定 catch 子句的 示例	208	14.4	一维数组和矩形数组	236
11.4.2	catch 子句段	209	14.5	实例化一维数组或矩形数组	236

14.6	访问数组元素	237	第 16 章 事件	269	
14.7	初始化数组	238	16.1	事件和委托相似	269
14.7.1	显式初始化一维数组	238	16.2	源代码组件概览	270
14.7.2	显式初始化矩形数组	239	16.3	声明事件	271
14.7.3	初始化矩形数组的语法点	239	16.3.1	事件是成员	271
14.7.4	快捷语法	240	16.3.2	委托类型和 EventHandler	272
14.7.5	隐式类型数组	240	16.4	触发事件	272
14.7.6	综合内容	241	16.5	订阅事件	273
14.8	交错数组	241	16.6	标准事件的用法	275
14.8.1	声明交错数组	242	16.6.1	使用 EventArgs 类	275
14.8.2	快捷实例化	242	16.6.2	通过扩展 EventArgs 来传递数据	276
14.8.3	实例化交错数组	242	16.6.3	使用自定义委托	276
14.8.4	交错数组中的子数组	243	16.7	MyTimerClass 代码	278
14.9	比较矩形数组和交错数组	244	16.8	事件访问器	279
14.10	foreach 语句	245	第 17 章 接口	280	
14.10.1	迭代变量是只读的	246	17.1	什么是接口	280
14.10.2	foreach 语句和多维数组	247	17.2	声明接口	283
14.11	数组协变	248	17.3	实现接口	284
14.12	数组继承的有用成员	249	17.4	接口是引用类型	285
14.13	比较数组类型	252	17.5	接口和 as 运算符	287
第 15 章 委托		253	17.6	实现多个接口	287
15.1	什么是委托	253	17.7	实现具有重复成员的接口	288
15.2	声明委托类型	255	17.8	多个接口的引用	289
15.3	创建委托对象	255	17.9	派生成员作为实现	291
15.4	赋值委托	257	17.10	显式接口成员实现	291
15.5	组合委托	257	17.11	接口可以继承接口	294
15.6	为委托增加方法	258	第 18 章 转换	297	
15.7	从委托移除方法	259	18.1	什么是转换	297
15.8	调用委托	259	18.2	隐式转换	298
15.9	委托的示例	260	18.3	显式转换和强制转换	299
15.10	调用带返回值的委托	261	18.4	转换的类型	300
15.11	调用带引用参数的委托	262	18.5	数字的转换	301
15.12	匿名方法	262	18.5.1	隐式数字转换	301
15.12.1	使用匿名方法	263	18.5.2	溢出检测上下文	302
15.12.2	匿名方法的语法	263	18.5.3	显式数字转换	303
15.12.3	变量和参数的作用域	265	18.6	引用转换	306
15.13	Lambda 表达式	266	18.6.1	隐式引用转换	307

18.6.2	显式引用转换	308
18.6.3	有效显式引用转换	308
18.7	装箱转换	309
18.8	拆箱转换	311
18.9	用户自定义转换	312
18.9.1	用户自定义转换的约束	312
18.9.2	用户自定义转换的示例	313
18.9.3	计算用户自定义转换	314
18.9.4	多步用户自定义转换的 示例	315
18.10	is 运算符	316
18.11	as 运算符	317
第 19 章	泛型	318
19.1	什么是泛型	318
19.2	C#中的泛型	320
19.3	泛型类	321
19.4	声明泛型类	321
19.5	创建构造类型	322
19.6	创建变量和实例	323
19.6.1	使用泛型的栈的示例	324
19.6.2	比较泛型和非泛型栈	325
19.7	类型参数的约束	326
19.7.1	Where 子句	327
19.7.2	约束类型和次序	327
19.8	泛型方法	328
19.8.1	声明泛型方法	328
19.8.2	调用泛型方法	329
19.8.3	泛型方法的示例	330
19.9	扩展方法和泛型类	331
19.10	泛型结构	332
19.11	泛型委托	332
19.12	泛型接口	334
19.12.1	使用泛型接口的示例	335
19.12.2	泛型接口的实现必须 唯一	336
19.13	泛型的协变和逆变	337
19.13.1	接口的协变和逆变	341
19.13.2	有关变化的更多内容	342
第 20 章	枚举数和迭代器	344
20.1	枚举数和可枚举类型	344
20.1.1	使用 foreach 语句	344
20.1.2	枚举数类型	345
20.2	使用 IEnumerator 接口	346
20.3	IEnumerable 接口	349
20.4	不实现接口的枚举数	351
20.5	泛型枚举接口	352
20.6	IEnumerator<T>接口	352
20.7	IEnumerable<T>接口	354
20.8	迭代器	355
20.8.1	迭代器块	356
20.8.2	使用迭代器来创建枚举数	357
20.8.3	使用迭代器来创建可枚举 类型	358
20.9	常见迭代器模式	360
20.10	产生可枚举类型和枚举数	360
20.11	产生多个可枚举类型	361
20.12	产生多个枚举数	362
20.13	迭代器实质	363
第 21 章	介绍 LINQ	365
21.1	什么是 LINQ	365
21.2	LINQ 提供程序	366
21.3	查询语法和方法语法	368
21.4	查询变量	369
21.5	查询表达式的结构	370
21.5.1	from 子句	371
21.5.2	join 子句	372
21.5.3	什么是联结	373
21.5.4	查询主体中的 from...let... where 片段	375
21.5.5	orderby 子句	378
21.5.6	select...group 子句	378
21.5.7	查询中的匿名类型	380
21.5.8	group 子句	380
21.5.9	查询延续	382
21.6	标准查询运算符	383
21.6.1	查询表达式和标准查询 运算符	384

21.6.2	标准查询运算符的签名	385	23.6	诊断指令	431
21.6.3	委托作为参数	386	23.7	行号指令	431
21.6.4	LINQ 预定义的委托类型	387	23.8	区域指令	432
21.6.5	使用委托参数的示例	388	23.9	#pragma warning 指令	433
21.6.6	使用 Lambda 表达式参数的示例	388			
21.7	LINQ to XML	390	第 24 章	反射和特性	434
21.7.1	标记语言	390	24.1	元数据和反射	434
21.7.2	XML 基础	390	24.2	Type 类	435
21.7.3	XML 类	391	24.3	获取 Type 对象	436
21.7.4	使用 XML 树的值	394	24.4	什么是特性	438
21.7.5	使用 XML 属性	397	24.5	应用特性	439
21.7.6	节点的其他类型	400	24.6	预定义的保留的特性	439
21.7.7	使用 LINQ to XML 的 LINQ 查询	401	24.6.1	Obsolete 特性	439
			24.6.2	Conditional 特性	440
			24.6.3	预定义的特性	441
第 22 章	异步编程简介	404	24.7	有关应用特性的更多内容	442
22.1	进程、线程以及异步编程	404	24.7.1	多个特性	442
22.1.1	多线程处理带来的问题	405	24.7.2	其他类型的目标	442
22.1.2	多线程处理的复杂度	405	24.7.3	全局特性	443
22.2	并行循环	406	24.8	自定义特性	443
22.3	BackgroundWorker 类	408	24.8.1	声明自定义特性	444
22.3.1	BackgroundWorker 类的示例代码	411	24.8.2	使用特性的构造函数	444
22.3.2	BackgroundWorker 用于 WPF 程序的例子	414	24.8.3	指定构造函数	444
22.4	异步编程模式	416	24.8.4	使用构造函数	445
22.5	BeginInvoke 和 EndInvoke	417	24.8.5	构造函数中的位置参数和命名参数	445
22.5.1	等待一直到结束模式	418	24.8.6	限制特性的使用	446
22.5.2	AsyncResult 类	419	24.8.7	自定义特性的最佳实践	447
22.5.3	轮询模式	420	24.9	访问特性	448
22.5.4	回调模式	421	24.9.1	使用 IsDefined 方法	448
22.6	计时器	423	24.9.2	使用 GetCustomAttribute 方法	449
第 23 章	预处理指令	426	第 25 章	其他主题	450
23.1	什么是预处理指令	426	25.1	概述	450
23.2	基本规则	426	25.2	字符串	450
23.3	#define 和 #undef 指令	427	25.2.1	使用 StringBuilder 类	451
23.4	条件编译	428	25.2.2	格式化数字字符串	452
23.5	条件编译结构	429	25.3	把字符串解析为数据值	455
			25.4	关于可空类型的更多内容	457

25.4.1 使用空接合运算符	458	25.6.2 使用其他 XML 标签	463
25.4.2 使用可空用户自定义 类型	459	25.7 嵌套类型	463
25.5 Main 方法	460	25.7.1 嵌套类的示例	464
25.6 文档注释	461	25.7.2 可见性和嵌套类型	465
25.6.1 插入文档注释	462	25.8 和 COM 的互操作	467



本章内容

- 在.NET之前
- 进入Microsoft .NET
- 编译成CIL
- 编译成本机代码并执行
- CLR
- CLI
- 缩写回顾

1.1 在.NET 之前

C#编程语言是为在微软公司的.NET框架^①上开发程序而设计的。本章将简要介绍.NET从何而来，以及它的基本架构。在开始之前，我要指出其正确发音：see sharp^②。

1.1.1 20 世纪 90 年代后期的 Windows 编程

在20世纪90年代后期，使用微软平台的Windows编程分化成许多分支。大多数程序员在使用 Visual Basic (VB)、C或C++。一些C和C++程序员在使用纯Win32 API，但大多数人在使用MFC (Microsoft Foundation Classes, 微软基础类库)。其他人已经转向了COM (Component Object Model, 组件对象模型)。

所有这些技术都有自己的问题。纯Win32 API不是面向对象的，而且使用它的工作量比使用MFC的更大。MFC是面向对象的，但是它却不一致，并逐渐变得陈旧。COM虽然概念简单，但它的实际代码复杂，并且需要很多丑陋的、不雅的底层基础代码。

所有这些编程技术的另外一个缺点是它们主要针对桌面程序而不是Internet进行开发。那时，Web编程还是以后的事情，而且看起来和桌面编程非常不同。

① 微软正式中文文献中一般称.NET Framework，本书考虑了国内读者习惯，统一译为.NET框架。——编者注

② 有一次我去应聘一个C#编程的职位，当时人力资源面试官问我从事“see pound”（应为see sharp）的经验有多少！我过了一会儿才弄清楚他在说什么。

1.1.2 下一代平台服务的目标

我们真正需要的是一个新的开始——一个集成的、面向对象的开发框架，它可以把一致和优雅带回编程。为满足这个需求，微软宣布开发一个代码执行环境和一个可以实现这些目标的代码开发环境。这些目标列在图1-1中。



图1-1 下一代平台的目标

1.2 进入 Microsoft .NET

在2002年，微软发布了.NET框架的第一个版本，声称其解决了旧问题并实现了下一代系统的目标。.NET框架是一种比MFC和COM编程技术更一致并面向对象的环境。它的特点包括以下几点。

- 多平台 该系统可以在多种多样的计算机上运行，从服务器、桌面机到PDA和移动电话。
- 行业标准 该系统使用行业标准的通信协议，比如XML、HTTP、SOAP和WSDL。
- 安全性 该系统能提供更加安全的执行环境，即使有来源可疑的代码存在。

1.2.1 .NET 框架的组成

.NET框架由三部分组成，如图1-2所示。^① 执行环境称为CLR（Common Language Runtime，公共语言运行库）。CLR在运行期管理程序的执行，包括以下内容。

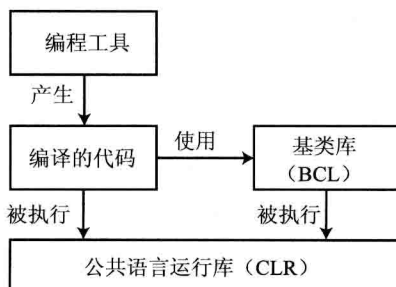


图1-2 .NET框架的组成

^① 严格地说，.NET框架由CLR和FCL（框架类库）两部分组成，不包括工具。FCL是BCL的超集，还包括Windows Forms、ASP.NET LINQ以及更多命名空间。——编者注