

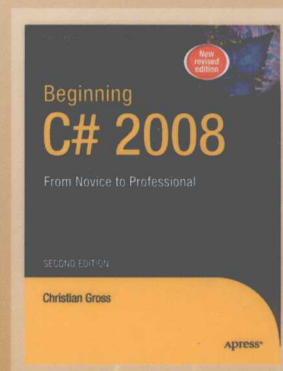
Beginning C# 2008 From Novice to Professional, Second Edition

C#基础教程

(第2版)

[美] Christian Gross 著
张骥 译

- C#初学者的入门图书
- 在问题求解中学习编程语言
- 理论和实践完美结合



TURING

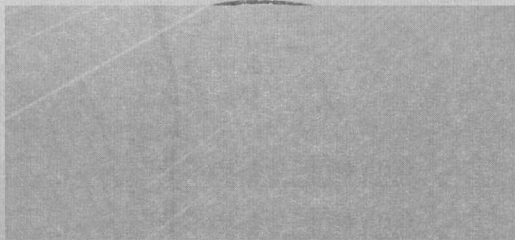
图灵程序设计丛书 微软技术系列

Beginning C# 2008 From Novice to Professional, Second Edition

C#基础教程

(第2版)

[美] Christian Gross 著
张骥 译



人民邮电出版社
北京

图书在版编目 (CIP) 数据

C# 基础教程: 第2版 / (美) 格罗斯 (Gross, C.)
著; 张骥译. —北京: 人民邮电出版社, 2010.3
(图灵程序设计丛书)
书名原文: Beginning C# 2008: From Novice to
Professional, Second Edition
ISBN 978-7-115-22282-4

I. ① C… II. ① 格…② 张… III. ① C 语言—程序设
计 IV. ① TP312

中国版本图书馆CIP数据核字 (2010) 第023824号

内 容 提 要

本书是一本 C# 入门图书, 共分 17 章, 除讲解了 C# 的语言基础、异常处理、面向对象及面向组件的基础知识外, 还介绍了 Lambda 表达式、持久化存储、.NET 泛型、应用程序配置和动态加载、多线程、关系型数据库、LINQ、函数式代码及 C# 的其他相关问题。

本书不仅适合没有任何编程语言基础的初级读者, 也是有 VB、C++ 等语言基础的 C# 初学者的极佳选择。

图灵程序设计丛书

C#基础教程 (第2版)

- ◆ 著 [美] Christian Gross
- 译 张 骥
- 责任编辑 傅志红
- ◆ 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>
- 北京鑫正大印刷有限公司印刷
- ◆ 开本: 800×1000 1/16
- 印张: 25
- 字数: 591千字 2010年3月第1版
- 印数: 1-3 000册 2010年3月北京第1次印刷
- 著作权合同登记号 图字: 01-2009-2891号

ISBN 978-7-115-22282-4

定价: 59.00元

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

反盗版热线: (010)67171154

前 言

我读的第一本计算机编程书是Charles Petzold写的*Programming Windows 3.0*。那大约是在1992年，微软凭借Windows 3.0已经向业界证明了它是一个充满前途的公司。当年，许多乱七八糟的事把为Windows写程序搞得十分复杂，文档缺乏、处理器只有16位，除了软件开发工具包之外还需要购买编译器。Charles的书将这些问题一并解决，只要看他这一本书，基本就可以为Windows写程序了。

现在人们面临的问题正好相反：文档太多了，处理器已经达到了64位，所有相关的东西都被整合到开发环境当中。现在的当务之急是要搞清楚我们到底需要什么。人们面临的选择实在太多了——一个问题就有多种解决途径。这本书想说明的与我刚开始学Windows编程时Charles教我的一样，即帮助读者弄清楚写代码到底需要用到什么。

这本书以解决问题为导向，讲解如何用C#编程。现在，C#已经发展成为一门复杂的编程语言，能实现众多功能，而你需要搞明白的是什么时候使用哪种方法。这本书解答了这些问题。

注意，本书不是包含C#各项功能的编程使用手册。那些生僻难懂的功能书中概不涉及，我讲的东西都是平时经常要用到的。放心，你不会漏掉那些C#语言构造，毕竟这本书已经包括了所有C#的主要功能。

想通过阅读这本书达到最好的效果，就应当完成每章的练习。练习的答案可以从Apress的网站<http://www.apress.com>网上下载^①。

如果你对C#一无所知，那么读完本书并做了练习后，我确信你就能成为基础扎实、深谙C#内涵的程序员。觉得我的这个承诺许得有点儿大发了，是不是？其实它还是挺靠谱儿的。本书的目的不就是让你熟悉C#编程语言，学会如何使用它的各项功能嘛，而设置的这些练习确保你实实在在地掌握了学到的这些东西。

这些练习都有相当的难度，想几分钟搞定是绝对不可能的。要知道，我是用了整整5天才全部完成的。

如果你有问题，比如设置某个练习的目的是什么？可以到Skype找我，我的ID是christianhgross，不过千万别给我打电话。咱们可以先通过文本交流，解决不了再通过语音沟通。当然，你也可以给我写邮件，我的E-mail地址是christianhgross@gmail.com。

最后，谢谢你选择本书，祝你好运。

^① 练习答案也可从图灵网站www.turingbook.com的本书主页下载。——编者注

目 录

第 1 章 预备, 坐稳, 出发!	1	2.1.1 计算器的构思	25
1.1 下载和安装工具	1	2.1.2 确定计算器开发的工作要点	26
1.2 选择应用程序类型	3	2.2 实现类库	28
1.3 创建项目和解决方案	3	2.2.1 编写Add()方法	30
1.4 创建Windows应用程序	4	2.2.2 编写代码来测试Add()方法	32
1.4.1 查看源代码	5	2.2.3 数值和数字类型方面的问题	36
1.4.2 重新命名解决方案	5	2.3 数字类型和值类型	38
1.4.3 保存解决方案	6	2.3.1 值类型和引用类型	38
1.4.4 运行Windows应用程序	6	2.3.2 CLR数字类型	39
1.4.5 让Windows应用程序向你打个 “招呼”	7	2.4 完成计算器的开发	42
1.4.6 给应用程序添加注释	10	2.5 需牢记的要点	43
1.5 在解决方案的用户控件之间导航	11	2.6 练习	43
1.6 创建控制台应用程序	13	第 3 章 字符串操作	45
1.6.1 给解决方案添加控制台应用程 序项目	14	3.1 翻译应用程序的构思	45
1.6.2 让控制台应用程序向你打个 “招呼”	14	3.2 构建Translator应用程序	46
1.6.3 设置启动项目	14	3.2.1 创建Translator类	46
1.6.4 运行控制台项目	15	3.2.2 问候语的翻译	47
1.7 创建类库	15	3.2.3 创建测试应用程序	47
1.7.1 给解决方案添加类库项目	15	3.2.4 职责方面的问题解答	48
1.7.2 转移功能	15	3.2.5 研究String类型	49
1.7.3 定义引用	16	3.2.6 解决多余空格的问题	53
1.7.4 调用类库的功能	16	3.2.7 字符串的引用	57
1.7.5 使用变量和常量	18	3.2.8 字符映射	58
1.8 .NET Framework的工作原理	20	3.3 对语言 and 文化的处理	59
1.9 需牢记的要点	22	3.3.1 在Windows下设置文化和语言	59
1.10 练习	22	3.3.2 数字的解析与处理	60
第 2 章 .NET 数字类型和值类型	24	3.3.3 文化的处理	62
2.1 软件开发的要点与构思	24	3.4 需牢记的要点	64
		3.5 练习	64
		第 4 章 数据结构、决策和循环	65
		4.1 深度优先搜索算法	65

2 目 录

4.2 实现用户定义的类型.....	68	多知识.....	129
4.2.1 声明结构和类.....	68	6.5.1 预处理器指令的更多细节.....	129
4.2.2 值类型的限制.....	69	6.5.2 属性作用域的更多细节.....	131
4.3 搜索算法的构思.....	74	6.5.3 abstract关键字的更多细节.....	131
4.4 编写深度优先搜索代码.....	75	6.6 需牢记的要点.....	133
4.4.1 数据结构的定义和实现.....	75	6.7 练习.....	133
4.4.2 定义算法测试.....	84		
4.4.3 实现深度优先搜索算法.....	88	第7章 组件和对象层级	134
4.4.4 运行深度优先搜索算法.....	94	7.1 基本的税收概念.....	134
4.5 需牢记的要点.....	95	7.2 税收应用程序的构思.....	135
4.6 练习.....	96	7.3 用构想编程.....	135
第5章 C#异常处理	97	7.3.1 使用C#接口描述构想.....	136
5.1 错误、异常和异常处理.....	97	7.3.2 理解继承和组件的工作原理.....	138
5.2 运行调试器.....	98	7.4 实现税收计算引擎.....	143
5.3 处理异常.....	99	7.4.1 定义接口.....	143
5.3.1 捕捉异常.....	99	7.4.2 实现税收计算引擎的基类.....	144
5.3.2 实现异常处理器.....	101	7.4.3 使用默认的实现.....	148
5.3.3 栈展开的防护工作.....	104	7.4.4 实现基本的税收计算.....	150
5.3.4 过滤异常.....	105	7.5 使用税收计算引擎的基本功能.....	151
5.4 编写异常安全代码.....	107	7.5.1 实现税收计算引擎并计算税收.....	151
5.4.1 编写防范性代码.....	107	7.5.2 使用税收计算引擎.....	155
5.4.2 使用默认的状态.....	109	7.6 继承和类型强制转换的更多知识.....	156
5.4.3 处理警示性错误.....	110	7.6.1 继承的更多细节.....	156
5.5 需牢记的要点.....	111	7.6.2 类型强制转换的更多细节.....	161
5.6 练习.....	111	7.7 需牢记的要点.....	161
		7.8 练习.....	162
第6章 面向对象编程的基础知识	112	第8章 面向组件的架构	163
6.1 货币差价.....	112	8.1 内核.....	163
6.2 构思货币兑换应用程序.....	113	8.2 构思照明应用程序.....	164
6.3 为货币兑换应用程序编写测试.....	113	8.3 构建内核.....	165
6.3.1 从结构性代码起步.....	114	8.3.1 定义接口.....	165
6.3.2 基类.....	114	8.3.2 实现内核.....	168
6.3.3 继承.....	115	8.3.3 将内核定义为接口而不是类.....	182
6.3.4 使用C#属性.....	116	8.4 构建完整的应用程序.....	183
6.3.5 继承和作用域修饰符.....	119	8.4.1 定义一些房间.....	183
6.3.6 处理验证.....	122	8.4.2 实例化PublicRoom和 PrivateRoom.....	184
6.3.7 完成基类.....	124	8.5 进一步学习私有类和对象初始化.....	185
6.4 编写活跃交易和酒店交易的货币换算.....	125	8.5.1 私有类.....	185
6.4.1 实现ActiveCurrencyTrader.....	125	8.5.2 使用内嵌数据类型的对象初始 化.....	186
6.4.2 实现HotelCurrencyTrader.....	127		
6.5 预处理器指令、属性和抽象方法的更			

8.6 需牢记的要点	187	10.5.1 实现GetHashCode()	239
8.7 练习	187	10.5.2 实现Equals()	241
第9章 列表、委托和 lambda 表达式	189	10.6 需牢记的要点	243
9.1 集合的管理	189	10.7 练习	243
9.1.1 C# 2.0之前的集合管理	189	第11章 .NET 泛型研究	244
9.1.2 C# 2.0之后的集合管理	193	11.1 为何要使用.NET泛型	244
9.2 问题代码案例	194	11.2 服务器端电子数据表的原理	247
9.2.1 使用委托	197	11.3 服务器端电子数据表的架构	249
9.2.2 使用匿名方法	201	11.3.1 设计架构	249
9.2.3 使用委托进行多路广播	202	11.3.2 定义服务器电子数据表接口	250
9.2.4 使用lambda表达式	203	11.4 实现服务器电子数据表	257
9.3 lambda表达式	205	11.4.1 在电子数据表中 使用lambda 表达式	263
9.3.1 创建算法	205	11.4.2 在不知道类型的情况下分配 状态	266
9.3.2 使用lambda表达式实现算法	206	11.4.3 重写ToString()功能	268
9.4 集合类型的更多知识	207	11.4.4 使用枚举器迭代数据	268
9.4.1 使用一般列表	207	11.5 使用电子数据表	269
9.4.2 使用键/值对列表	208	11.5.1 计算平均值	269
9.4.3 使用Stack	209	11.5.2 计算的工作原理	271
9.4.4 使用Queue	209	11.6 需牢记的要点	272
9.5 需牢记的要点	210	11.7 练习	272
9.6 练习	210	第12章 应用程序配置和动态加载	273
第10章 关于持久化存储的所有相关 问题	211	12.1 惯例优于配置	273
10.1 构思彩票预测系统	211	12.1.1 使用配置架构解耦合	274
10.2 使用控制台灌入数据	212	12.1.2 使用惯例架构解耦合	275
10.2.1 从控制台读取数据	212	12.2 设置动态加载项目	275
10.2.2 构建外壳	213	12.2.1 给程序集签名	276
10.2.3 实现TextProcessor应用 程序	222	12.2.2 设置输出路径	277
10.3 灌入二进制数据	229	12.3 定义和处理配置文件	278
10.3.1 定义接口并实现外壳	230	12.3.1 创建基于XML的配置文件	279
10.3.2 定义类型	233	12.3.2 添加动态加载的配置项	280
10.3.3 把文本流转换成二进制流	234	12.3.3 读取配置文件	281
10.3.4 把二进制流转换成文本流	235	12.4 程序集的动态加载	281
10.4 调整序列化	237	12.4.1 类型的动态实例化	281
10.4.1 执行自定义序列化	237	12.4.2 增强配置文件	285
10.4.2 将数据成员声明为不可序 列化	238	12.5 加载强命名的程序集	289
10.4.3 分离数据对象和动作对象	238	12.5.1 将强命名的程序集重定位到 GAC	290
10.5 完成自定义类型	238	12.5.2 使用版本号	292

12.6	实现基于惯例的架构.....	294	14.5.2	使用生成的代码.....	341
12.7	动态加载基类或接口类型.....	296	14.6	需牢记的要点.....	342
12.8	需牢记的要点.....	296	14.7	练习.....	343
12.9	练习.....	297	第 15 章	学习 LINQ.....	344
第 13 章	关于多线程.....	298	15.1	找到中奖号码的频率.....	344
13.1	多任务处理.....	298	15.1.1	扩展彩票预测系统.....	345
13.1.1	抢占式多任务处理.....	299	15.1.2	实现数字频率解决方案.....	348
13.1.2	时间分割.....	299	15.2	学习更多的LINQ技巧.....	353
13.2	使用线程.....	301	15.2.1	选择和更改数据.....	356
13.2.1	创建新的线程.....	301	15.2.2	用匿名类型进行选择.....	357
13.2.2	等待线程结束.....	302	15.2.3	处理多个流.....	358
13.2.3	创建带状态的线程.....	303	15.2.4	给结果排序.....	358
13.2.4	线程之间的同步.....	304	15.2.5	给结果分组.....	359
13.2.5	如何防止代码死锁.....	309	15.2.6	执行集合运算.....	360
13.3	实现读/写线程架构.....	312	15.3	在其他环境中使用LINQ.....	363
13.4	实现生产者/消费者架构.....	315	15.4	需牢记的要点.....	364
13.4.1	使用隐藏的生产者/消费者 实现.....	316	15.5	练习.....	364
13.4.2	实现通用的生产者/消费者 架构.....	317	第 16 章	在 C#中编写函数式代码.....	365
13.4.3	使用异步法.....	319	16.1	为何要使用函数式编程.....	365
13.5	需牢记的要点.....	320	16.2	函数式编程的要领.....	367
13.6	练习.....	321	16.2.1	高阶函数.....	367
第 14 章	使用关系型数据库.....	322	16.2.2	纯函数.....	370
14.1	关系型数据库.....	322	16.2.3	函数求值.....	374
14.1.1	关系型数据库表.....	322	16.2.4	递归.....	376
14.1.2	数据库关系.....	323	16.3	需牢记的要点.....	377
14.2	访问关系型数据库.....	326	16.4	练习.....	377
14.3	使用 Visual C# Express 设计数据库.....	327	第 17 章	C#拾遗补缺.....	378
14.3.1	配置数据源.....	328	17.1	操作符.....	378
14.3.2	添加表.....	330	17.1.1	使用算术操作符.....	378
14.4	使用 ADO.NET 访问数据库.....	333	17.1.2	重载操作符.....	382
14.4.1	连接数据库.....	333	17.2	goto 语句.....	384
14.4.2	添加表数据.....	334	17.3	.NET 泛型约束.....	385
14.4.3	从表中选择数据.....	336	17.3.1	使用 type 约束.....	385
14.4.4	从数据库删除数据.....	336	17.3.2	使用 new 约束.....	386
14.4.5	关闭数据库连接.....	337	17.3.3	使用 class 约束.....	387
14.4.6	ADO.NET 用法提要.....	337	17.4	可空类型.....	387
14.5	使用 Dataset Designer.....	337	17.5	分部类和分部方法.....	389
14.5.1	构建表之间的关系.....	337	17.6	需牢记的要点.....	390
			17.7	练习.....	391

第1章

预备，坐稳，出发！



这是一本关于C#编程语言的书，它旨在帮助大家成为技艺精湛的C#程序员，即使你以前从未编过程序或者只使用过Visual Basic这样的过程语言。[C#被称为面向对象语言(object-oriented language)，其用法不同于Visual Basic、Pascal、COBOL和其他很多无人再用的过程语言(procedural languages)。]面向对象语言不仅是未来之趋势，而且是今日之潮流。如果不知道如何使用Java、C++或C#这样的面向对象语言，我们就无法为Web编程。如果想使用.NET平台编写网站并实现Web数据交换（这是一种极其流行的做法），那就要学习C#这门语言。

在这一章，大家首先要获得开发C#应用程序所需要的工具，然后用这些工具小试一把。在这个过程中，我们会创建几个C#应用程序。

1.1 下载和安装工具

如果是初次使用C# 3.0，你可能会热切地想要编写一些能够做实事的代码。.NET的妙处就在于：安装好.NET软件开发工具包(.NET SDK)或Visual Studio集成开发环境(IDE)之后，我们立即就可以开始编写代码了。下载并安装正确的环境，是迈向富有成效且有价值的编码体验的至关重要的第一步。

说明 本书所讲解的是针对.NET Framework的应用程序编程语言C# 3.0。对于C# 3.0，我们将使用.NET Frameworks 3.0和3.5。.NET 3.0为我们提供了编程和编码所需的全部要素，.NET 3.5则为我们提供了许多附加功能和更多编程选择。

对于本书中的示例，我们将使用Visual C# 2008 Express Edition。为什么？因为它是免费的，而且它拥有C# 3.0入门所需要的一切。微软还有其他一些Express Edition IDE，针对的是不同的语言（Visual Basic和C++）或支持特定的功能（如Visual Web Developer Express）。对我们来说，这些版本太过局限。

微软还提供了Visual Studio IDE的一些完整版本，如Standard、Professional和Team版。每个版本的功能和定价各不相同。详细信息，请参见Microsoft Visual Studio网站（<http://msdn2.microsoft.com/en-us/vstudio/default.aspx>）。如果已经安装了Visual Studio 2008 Professional，那就可以使用它来完成本书中的示例。Visual C# Express能做的事情，Visual Studio 2008全都可以做。事实上，后者具有更多的选择。

说明 我个人是将Visual Studio Standard或Visual Studio Professional与其他一些工具结合起来使用, 如Omnicoe公司的X-develop和JustCode! (<http://www.omnicore.com>)、TestDriven.NET (<http://www.testdriven.net/>) 和NUnit (<http://www.nunit.org>)。虽然Visual Studio产品非常优秀, 但是也有其他一些产品可以用。作为一名优秀的开发人员, 就应该知道哪些工具可用并懂得判断哪些工具是最适合自己的。

从微软的网站安装和下载Visual C# Express会牵涉到大体积文件的传输。如果没有宽带连接, 最好用CD来安装IDE, 我们可以从微软公司的在线网站订购CD。

下载 Visual C# Express

下面是从微软的网站下载Visual C# Express的操作步骤。到大家阅读本书的时候, 实际的操作步骤可能有所变化, 但找到并下载IDE数据包的整个过程应该大致相似。

- (1) 前往<http://msdn.microsoft.com/vstudio/express/>。
- (2) 选择Visual Studio 2008 Express Editions链接。
- (3) 选择Windows Development (因为这正是我们在这本书中要做的工作)。
- (4) 单击Visual Studio Express Download链接。

(5) 我们会看到一个Visual Studio Express版本列表, 如图1-1所示。单击Visual C# 2008 Express Edition。

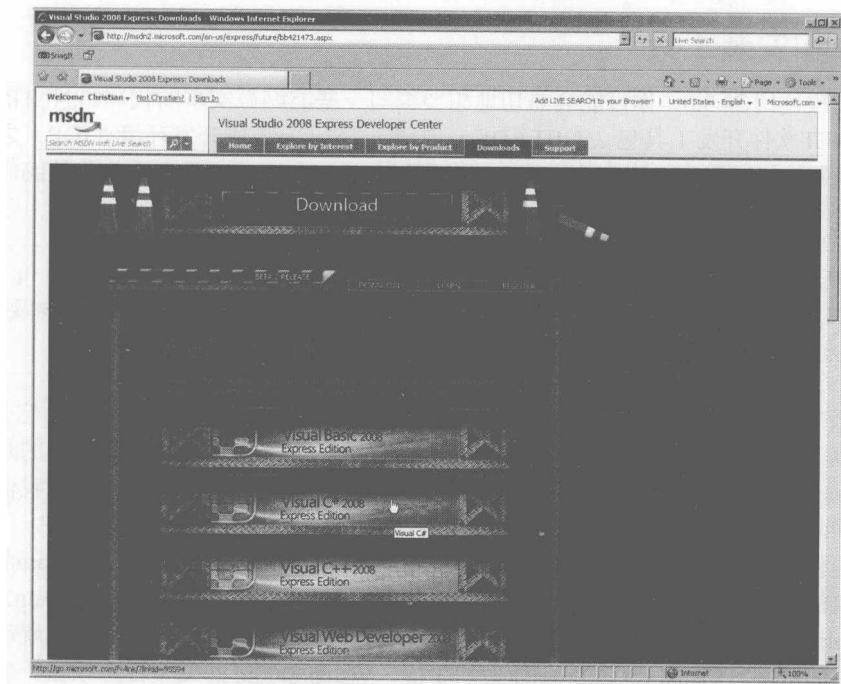


图1-1 选择Visual C# 2008 Express Edition

(6) 此时会出现一个对话框，询问我们要把下载的文件存放在哪里。此时下载的是一个小的引导文件，我们将使用它开始真正的Visual C# Express IDE的安装。选择将文件保存在桌面上。

这些步骤执行起来很快——可能几分钟就能完成。如果大家遵循这种做法，请不要误以为该过程是在下载完整的Visual C# Express应用程序，因为事实并非如此。大部分的IDE会在安装过程（即我们接下来将要执行的操作）中下载。此时，我们下载的只是初始的安装文件。

下载了安装文件之后，就可以开始安装Visual C# Express了。在这个过程中，IDE的所有组件（约300MB）都会下载并安装。请按照下列步骤进行操作。

(1) 双击桌面上的vcssetup.exe文件。等待安装程序加载所有必需的组件。

(2) 单击初始安装界面上的Next按钮。

(3) 此时会出现一系列对话框。选择默认的设置，单击Next按钮，继续运行安装程序。在最后的对话框中，单击Install按钮。

(4) 下载并安装好所有组件之后，可能需要重新启动计算机。

安装好Visual C# Express之后，就可以从“开始”菜单中选择并启动它了。

1.2 选择应用程序类型

Visual C# Express运行起来之后，我们就做好编写首个.NET应用程序的准备了。不过，我们需要首先做一个选择：将要编写的应用程序是什么类型的？概括地说，在.NET中，我们可以开发如下3种主要类型的程序。

- 在没有用户界面的命令行中运行的控制台应用程序（console application）。
- 在用户的桌面上运行并具有用户界面的窗口应用程序（Windows application）。
- 包含许多可重用功能的类库（class library），这些功能可以由控制台和Windows应用程序使用。这个库本身是无法运行的。

现在，大家对每种类型的程序都有了大致的了解。在本章，我们会编写全部3种类型的程序。它们是不同的“Hello, World”示例，都会在屏幕上显示文本“Hello, World”。使用“Hello, World”程序示范编程语言的功能已经有几十年的历史了。

1.3 创建项目和解决方案

在使用Visual Studio系列产品的时候，不论要编写的程序是哪种类型的，都要创建项目和解决方案。

- 项目（project）这个类别，用来表示某种类型的.NET应用程序。
- 解决方案（solution）这个类别，用来表示相互关联的多个.NET应用程序。

以制造汽车为例。转向系统可以是一个项目，排放系统可以是另一个项目，启动系统又可以是一个项目。把所有汽车项目放在一起就会创建一个名为“汽车”的完整解决方案。

其要点就是：解决方案包含多个相关的项目。对于本章的示例，我们的解决方案将包含3个项目，这些项目分别代表3种不同的程序类型。

在使用Visual C# Express的时候，创建项目就意味着创建解决方案，因为创建一个没有项目的空解决方案是没有意义的。那就像是构造一台没有任何零件的汽车。当书中提到“项目”或“应用程序”的时候，从工作空间的组织角度来看，它们表示的是一回事儿。解决方案则明确表示一

个或多个项目或应用程序。

本章对项目 and 解决方案方面的操作规划如下。

- 通过创建名为Example1的Windows应用程序创建一个.NET解决方案(创建该应用程序的同时也会创建一个解决方案)。
- 为所创建的解决方案添加一个名为Example2的控制台应用程序。
- 为所创建的解决方案添加一个名为Example3的类库项目。

1.4 创建 Windows 应用程序

我们现在就做, 从创建Windows应用程序开始。运行了Visual C# Express之后, 可以按照下列步骤创建Windows应用程序。

(1) 从菜单中选择File → New Project命令。

(2) 选择Windows Application图标。这个图标所代表的项目类型是基于一个名为Windows Application的预定义模板。

(3) 将默认名称改为Example1。

(4) 单击OK按钮。

这些操作步骤会在创建新项目的同时创建解决方案, 即Example1解决方案和Example1项目。Visual C# Express会显示出完整的项目和解决方案, 如图1-2所示。

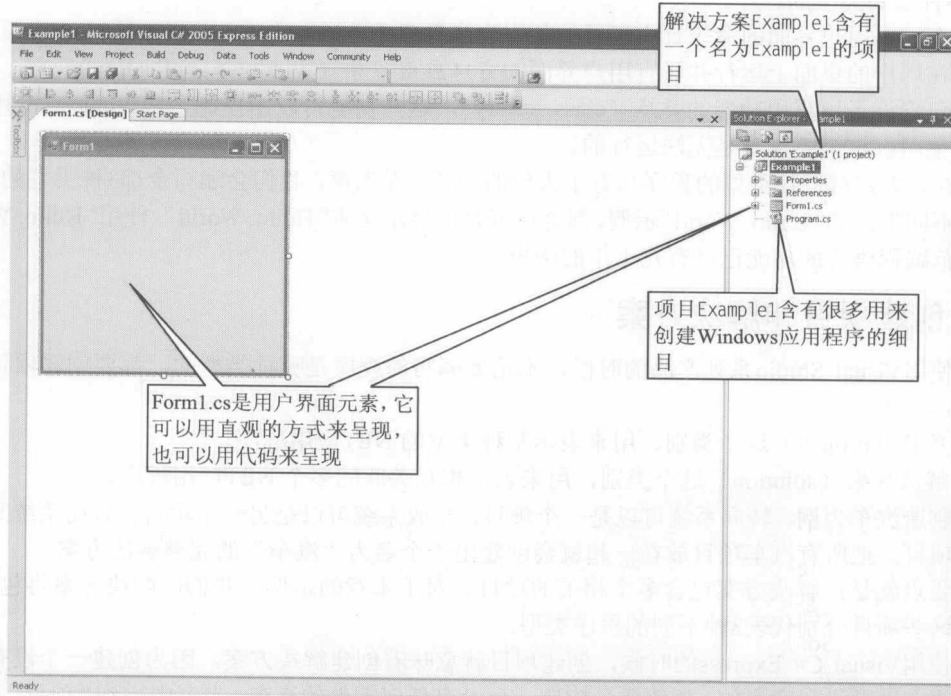


图1-2 显示出新的Example1项目及解决方案的Visual C# Express IDE

1.4.1 查看源代码

在创建新应用程序的时候，Visual C# Express会为其自动生成一些源代码。双击Solution Explorer中的Program.cs，看看所生成的代码。图1-3中显示的源代码会出现在Solution Explorer左侧的区域中。

说明 若想在用户界面和所生成的代码之间切换，可用右键单击Solution Explorer中的Form1.cs。这时会出现一个子菜单，显示有View Code（查看代码）和View Designer（查看用户界面）选项。

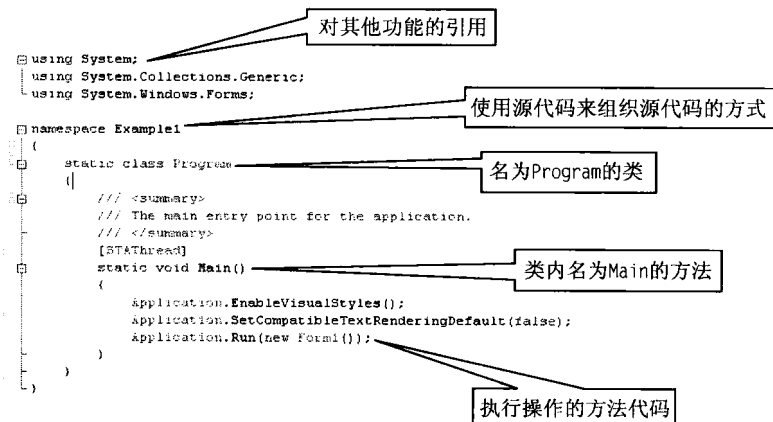


图1-3 C#文件中的源代码片段

图1-3中的标注文字表明了所要编写的C#源代码的实质。对这些内容的学习将贯穿整本书。目前，要记好的主要概念如下。

- **类**：将相关代码聚集在一起的组织单元。其专用程度比解决方案或项目的专用程度更高。再次以汽车为例，如果引擎是一个项目，那么起动机系统就是一个类。另一个类可能是排放系统。还有一个类可能是转向系统。换句话说，项目是由多个类组成的。
- **方法**：完成某项任务的一组指令。方法就相当于其他很多语言中的函数。Main()方法会在应用程序启动的时候运行，因此，它包含的是我们想要在程序运行之初使用的代码。

1.4.2 重新命名解决方案

Visual C# Express会自动将解决方案和项目命名为Example1，但这并非理想的名称。好在，重新命名解决方案是件很容易的事情。请按照下列步骤进行操作。

- (1) 右键单击Solution Explorer中的解决方案名称，从右键菜单中选择Rename命令。
- (2) 此时，解决方案名称会变成可编辑的状态，将其改为ThreeExamples。
- (3) 按Enter键应用所做的更改。

可以使用相同的方法, 给项目或显示在Solution Explorer中的其他元素重新命名。

1.4.3 保存解决方案

重新命名解决方案之后, 最好保存所做的更改。若想保存项目, 请按照下列步骤操作。

(1) 在Solution Explorer中突出显示解决方案的名称。

(2) 选择File → Save ThreeExamples.sln命令。

(3) 注意, Visual C# Express会使用旧的名称Example1, 而不是新的解决方案名称(Three-Examples)保存解决方案。若想将新的解决方案名称保存到硬盘上, 需要再次将Example1改成ThreeExamples。注意Visual C# Express保存项目的路径——我们将时不时地提到这个路径。

(4) 单击Save按钮。

成功保存了解决方案和项目之后, 我们会在窗口左下角的状态栏中看到“Item(s) Saved”这条消息。

以后, 我们可以随时使用键盘快捷键Ctrl+S来保存解决方案和项目。

说明 如果尚未保存所做的更改, 那么在选择退出Visual C# Express的时候, 系统就会询问是否要保存或丢弃解决方案及项目。

若想打开之前保存的解决方案, 可以随时选择File → Open Project命令, 导航至解决方案文件。一开始启动Visual C# Express的时候, 我们还可以从Recent Projects窗口中选择解决方案。而且, Recent Projects窗口会始终位于主Visual C# Express窗口的Start Page选项卡上。

1.4.4 运行 Windows 应用程序

由Visual C# Express生成的源代码是一个基本应用程序, 该程序包含一个不带任何功能的空白窗口。换句话说, 源代码只是为我们提供一个起点。有了这些源代码, 我们可以根据需要添加更多的源代码来创建自己的解决方案, 可以调试已经写好的源代码, 当然还可以使用源代码来运行和测试应用程序。

若想运行应用程序, 可选择Debug → Start Without Debugging命令。我们还可以使用键盘快捷键Ctrl+F5来完成相同的任务。当应用程序启动的时候, 我们会看到一个窗口, 其中显示了应用程序的代码——在这个案例中, 为Example1应用程序。通过单击该窗口的关闭按钮, 可以退出应用程序。图1-4说明了这一过程。

运行应用程序使我们能够看到它所做的工作。通过IDE运行应用程序, 就等同于用户通过从桌面单击来启动应用程序。在这个示例中, Example1会显示出一个没有任何控件或功能的空窗口。源代码的功能就是在启动的时候显示一个空白窗口, 并提供一个结束应用程序的按钮。

此时, 我们一行代码还未编写就创建了一个应用程序, 并且真的有所显示, 这一切都是因为Visual C#会生成一些即刻可用的样板C#代码。我们刚才创建了一个应用程序, 看到了它的源代码, 并运行了它。所有工作都是在这个名为Visual C# Express的舒适、体贴的开发环境下完成的。

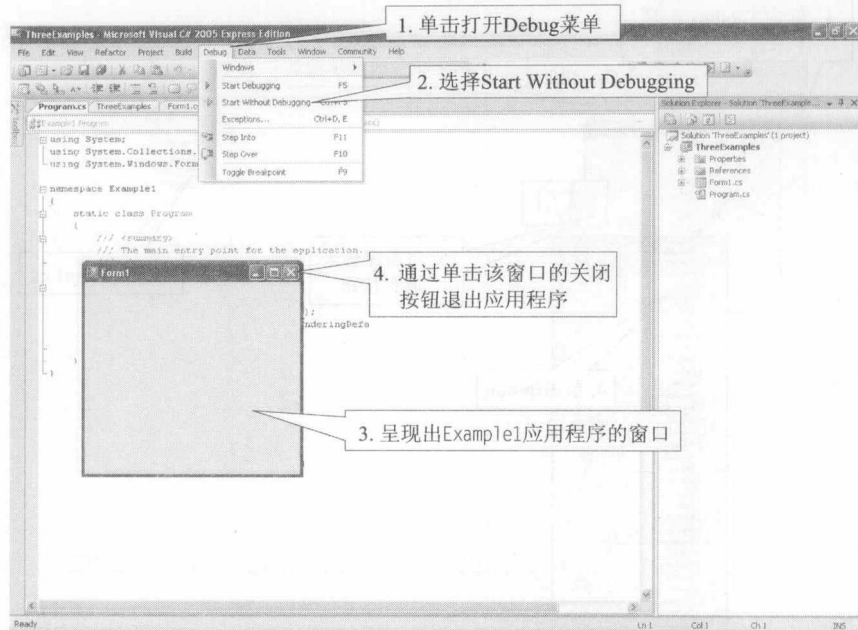


图1-4 运行应用程序

Visual C# Express有好处也有坏处。Visual C# Express的优势，是它隐藏了杂乱的细节，但细节被隐藏也正是它的弊端之所在。不妨想象自己是一个汽车机修工，试想一下你获得的诊断信息非常之少以至于工作难以展开。汽车制造商生产的仪表板上有一些灯，这些灯会在汽车出故障的时候亮起来。对于驾驶者来说，这是好东西。他会知道何时该把汽车带你这里来。但对于机修工来说，这就不够好。虽然闪烁的灯会用信号告诉我们出现了问题，但它并没有指明问题究竟是什么。这可不好。所以，我们来看看如何将它的优缺点加以利用。

1.4.5 让 Windows 应用程序向你打个“招呼”

我们创建的Windows应用程序现在只是显示了一个可以关闭的空白窗口，除此之外，没有其他功能。要想使应用程序做些什么，就需要添加用户界面元素或添加一些代码。只添加代码而不添加用户界面元素虽然可以让程序具备功能，但得到的结果对用户不是非常友好。所以，我们要添加一个按钮，单击这个按钮就会在文本框中显示“hello, world”。

首先，我们需要给窗体添加一个Button控件。双击Solution Explorer中的Form1.cs，显示出一个空白窗体。然后单击Toolbox选项卡，访问控件。单击Button，然后单击窗体，把按钮放在窗体上。图1-5给出了这些步骤。

接下来，我们将使用相同的基本步骤添加一个TextBox控件。最后，将按钮和文本框对齐，如图1-6所示。若想移动控件，可使用在突出显示控件时所出现的手柄。在拖动控件的时候，Visual C# Express会让控件的边缘“吸”到最近的几何边缘上。这种实用的支持使我们能够更加精确地对齐控件。

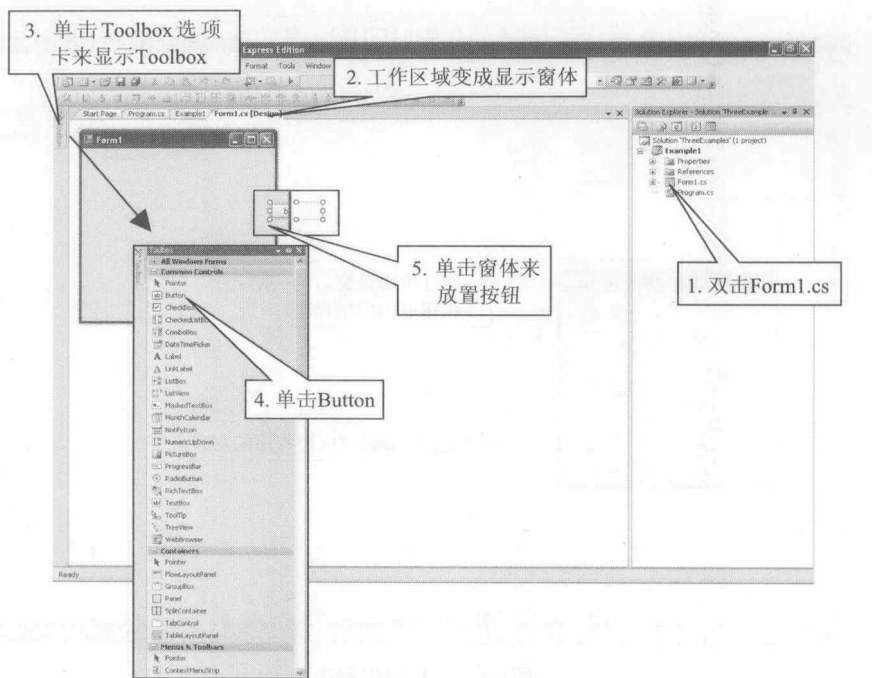


图1-5 给窗体添加按钮

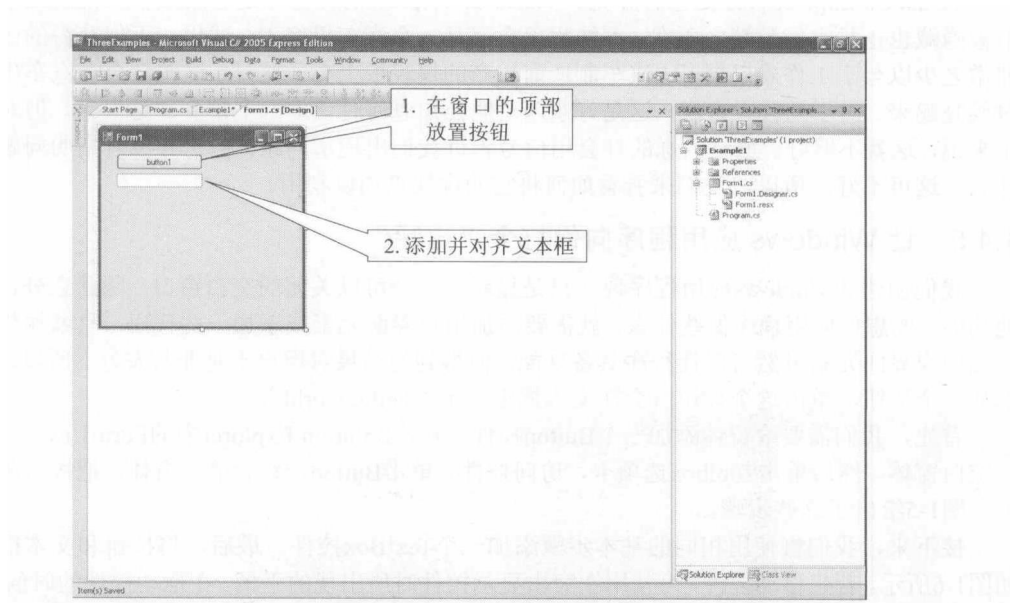


图1-6 对齐之后的按钮和文本框

如果通过按Ctrl+F5键执行Example1, 应该就会看到一个窗口, 其中包含了按钮和文本框, 如图1-6所示。此时可以单击按钮, 并且可以在文本框中添加文本或从中删除文本。但不管执行哪个操作, 现在都没有实际效果, 因为没有与这两个控件关联的代码。此时的按钮和文本框只是静态的用户界面元素。

若想让应用程序做些什么, 就需要在事件(event)方面有所考虑。举个例子, 如果有一个带自动开门器的车库, 你会希望按下远程控制按钮就可以在车库门关着的时候将其打开, 而在车库门开着的时候将其关闭。在Example1中, 我们要把按钮的单击与在文本框中显示文本的动作关联起来。

选择窗体上的按钮, 双击它。此时的工作区域变成了源代码视图, 光标位于button_Click函数内。添加下面这行源代码:

```
TextBox1.text = "hello, world";
```

图1-7说明了将事件与动作关联起来的操作过程。

注意, 添加到窗体中的文本框的名称是textBox1。这个名称是由Visual C# Express生成的, 正如按钮的默认名称也是由其生成的那样。我们可以(通过各个控件的Properties窗口)更改默认的名称, 但对于这个示例, 我们保留默认的名称不变。

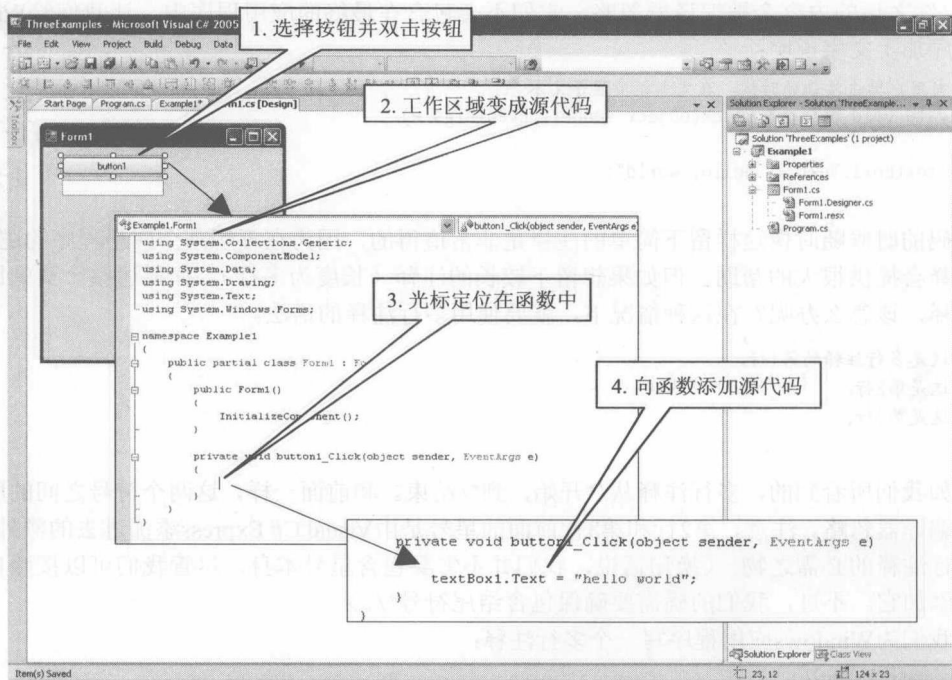


图1-7 将单击按钮事件与向文本框中添加文本的动作关联起来

给事件添加动作非常简单, 只要按照图1-7中所示的步骤操作即可。这种简易性要归功于