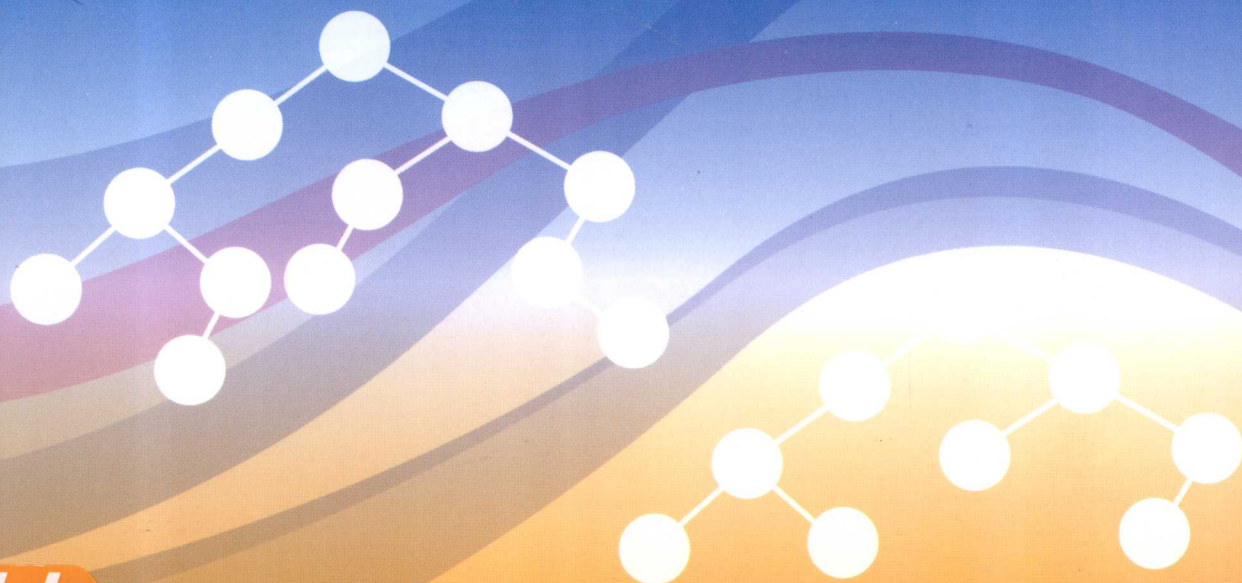


高等学校计算机专业规划教材

Visual C#程序设计 (2012版)



邱仲潘 王帅 孙赫雄 编著

清华大学出版社

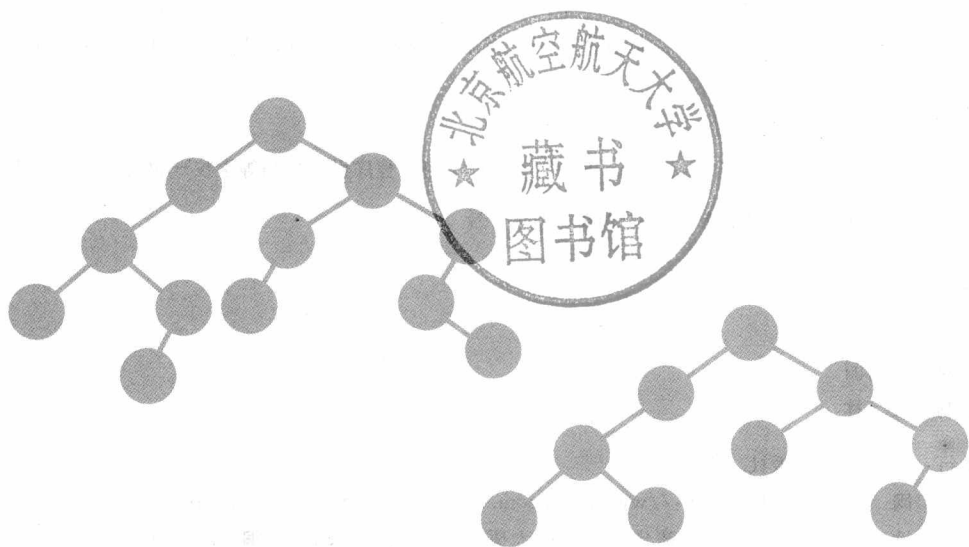


013034451

TP312C
2168

高等学校计算机专业规划教

Visual C#程序设计 (2012版)



邱仲潘 王帅 孙赫雄 编著

TP312C
2168

清华大学出版社



北航 C1641759

123380810

内 容 简 介

本书以 Visual C# 2012 语言为工具,介绍面向对象程序设计中的基本概念和方法,并通过大量的程序实例和相关练习逐步掌握高级程序设计语言的基本知识和基本技术,在理论和实践上使学生掌握面向对象的思想方法并初步具备软件开发的能力。全书共 17 章,内容包括流程控制语句、类与对象、继承与多态、委托与事件、数组与集合、泛型、反射、字符串操作、文件与流、多线程编程、Windows 窗体与控件设计、数据库编程、网络通信编程、异步编程等。本书的代码可以直接使用在读者开发的应用程序中,非常适合作为 Visual C# 程序设计的教材或参考用书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

Visual C# 程序设计:2012 版 / 邱仲潘,王帅,孙赫雄编著. —北京:清华大学出版社,2013.4
高等学校计算机专业规划教材
ISBN 978-7-302-30987-1

I. ①V… II. ①邱… ②王… ③孙… III. ①BASIC 语言—程序设计—高等学校—教材 IV. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 301993 号

责任编辑:龙启铭

封面设计:何凤霞

责任校对:焦丽丽

责任印制:宋 林

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座 邮 编:100084

社 总 机:010-62770175 邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

课 件 下 载: <http://www.tup.com.cn>, 010-62795954

印 刷 者:清华大学印刷厂

装 订 者:三河市新茂装订有限公司

经 销:全国新华书店

开 本:185mm×260mm 印 张:34

字 数:783 千字

版 次:2013 年 4 月第 1 版

印 次:2013 年 4 月第 1 次印刷

印 数:1~3000

定 价:69.00 元

产品编号:050226-01



编程语言这么多,为什么选择C#?因为C#语言是一门简单、现代、优雅、面向对象、类型安全、平台独立的新型组件编程语言。其语法风格源自大家熟悉的C/C++家族,融合了Visual Basic的高效和C/C++强大,其优雅的语法风格、创新的语言特性深受世界各地程序员的好评和喜爱。众所周知,C#起源于C语言家族,C/C++和Java的程序员能很快熟悉它。C#获得了ECMA和ISO/IEC的国际标准认证,分别是ECMA-334标准和ISO/IEC 23270标准。.NET框架的C#编译器就是根据这两个标准实现的。如果你希望学习微软公司最新推出的Visual C# 2012,请试一试这本书。本书作者从事IT行业二十多年,是最早一批接触C#的人,而且至今仍然在这块土地上辛勤耕耘。本书一定能够给你一段轻松愉快的学习历程。

本书的作者包括邱仲潘、王帅、孙赫雄,宋智军、张星成、黄宇轩也参与了其中,做了大量工作,在此深表感谢。



第 1 章 Visual C# 2012 简介 /1

1.1 C# 概述	1
1.1.1 C# 的产生与特点	1
1.1.2 C# 4.0 新特性	2
1.2 .NET Framework 概述	8
1.2.1 .NET Framework 简介	8
1.2.2 C# 与 .NET Framework 的关系	8
1.2.3 .NET Framework 4.5	9
1.3 Visual Studio 2012 概述	11
1.3.1 Visual Studio 2012 的新特性	11
1.3.2 Visual Studio 2012 的安装环境与安装过程	14
1.3.3 认识 Visual Studio 2012 集成开发环境	17

第 2 章 第一个 Visual C# 2012 程序 /21

2.1 编写 Hello World 程序	21
2.2 代码详解	23
2.2.1 程序结构	24
2.2.2 命名空间	24
2.2.3 using 语句	26
2.2.4 声明类	28
2.2.5 声明 Main 方法	28
2.2.6 程序语句	29
2.2.7 程序代码注释	30
2.3 程序调试	33
2.3.1 程序错误类型	33
2.3.2 程序调试方法	34
2.4 发布程序	36

第 3 章 Visual C# 2012 编程基础 /39

3.1 语句	40
--------------	----



3.2	标识符	40
3.3	关键字	42
3.4	数据类型	43
3.4.1	值类型	43
3.4.2	引用类型	50
3.5	类型转换	56
3.5.1	隐式类型转换	56
3.5.2	显式类型转换	59
3.5.3	装箱	63
3.5.4	拆箱	63
3.6	常量和变量	64
3.6.1	常量	65
3.6.2	变量	65
3.6.3	变量的命名规则	66
3.6.4	变量类型	67
3.6.5	变量作用域	68
3.7	运算符	71
3.7.1	算术运算符	71
3.7.2	关系运算符	72
3.7.3	逻辑运算符	73
3.7.4	位运算符	74
3.7.5	赋值运算符	75
3.7.6	其他运算符	76
3.7.7	运算符优先级	80
3.8	运算符重载	82
3.8.1	运算符工作方式	82
3.8.2	运算符重载示例	83
3.9	预处理器指令	85
3.9.1	#define 和 #undef	85
3.9.2	#if、#elif、#else 和 #endif	86
3.9.3	#warning 和 #error	87
3.9.4	#region 和 #endregion	88
3.9.5	#line	89
3.9.6	#pragma、#pragma warning 和 #pragma checksum	91

第 4 章 流程控制语句 /94

4.1	选择语句	94
4.1.1	if 语句	94



4.1.2	switch 语句	97
4.2	循环语句	98
4.2.1	while 语句	99
4.2.2	do-while 语句	99
4.2.3	for 语句	101
4.2.4	foreach 语句	102
4.3	跳转语句	103
4.3.1	break 语句	103
4.3.2	continue 语句	104
4.3.3	goto 语句	105
4.3.4	return 语句	107
4.4	异常处理语句	108
4.4.1	throw 语句	110
4.4.2	try-catch 语句	112
4.4.3	try-finally 语句	115
4.4.4	try-catch-finally 语句	116

第 5 章 类与对象 /118

5.1	面向对象编程	118
5.2	类	119
5.2.1	类的定义	119
5.2.2	类的访问修饰符	120
5.2.3	类的成员	122
5.2.4	成员的访问修饰符	123
5.2.5	类的实例	126
5.3	常量	127
5.3.1	静态常量	127
5.3.2	动态常量	129
5.4	字段	131
5.5	属性	132
5.5.1	属性的声明	133
5.5.2	属性修饰符	134
5.5.3	属性与字段的区别	136
5.6	构造函数和析构函数	137
5.6.1	构造函数	137
5.6.2	析构函数	143
5.7	方法	144
5.7.1	方法的声明与调用	144



5.7.2	方法的参数	146
5.7.3	静态方法与非静态方法	151
5.7.4	外部方法	151
5.7.5	方法重载	152
5.8	索引器	154
5.9	类与结构的比较	156
第 6 章	继承与多态	/160
6.1	什么是继承	160
6.2	基类和派生类	161
6.3	与继承相关的关键字	163
6.3.1	base 关键字——基类成员的访问	163
6.3.2	new 关键字——基类成员的隐藏	164
6.3.3	virtual 关键字和 override 关键字——虚拟与实现	166
6.3.4	virtual 关键字与 new 关键字的区别	168
6.3.5	abstract 关键字——抽象类	170
6.3.6	sealed 关键字——密封类	172
6.4	接口	174
6.4.1	接口的定义	174
6.4.2	接口的成员及其全权名	175
6.4.3	接口成员的访问及其二义性	177
6.4.4	接口的实现	179
6.5	抽象类与接口	181
6.6	多态性	185
第 7 章	委托与事件	/186
7.1	委托	186
7.1.1	委托的声明	186
7.1.2	委托的实例化及调用	187
7.1.3	多播委托	189
7.1.4	委托中的协变与逆变	192
7.1.5	委托与接口	194
7.2	事件	195
7.2.1	事件的声明	196
7.2.2	事件访问器	198
7.2.3	事件的订阅与取消	199
7.2.4	在派生类中引发基类事件	204
7.2.5	实现接口事件	207



7.2.6 使用字典存储事件实例.....	211
-----------------------	-----

第 8 章 数组与集合 /215

8.1 数组概述	216
8.2 数组的种类	217
8.2.1 一维数组.....	217
8.2.2 多维数组.....	219
8.2.3 交错数组.....	221
8.3 数组元素的访问	224
8.4 数组类	225
8.5 数组的基本操作	226
8.5.1 数组的遍历.....	226
8.5.2 数组的清空.....	227
8.5.3 数组的查找.....	229
8.5.4 数组的排序.....	231
8.6 数组作为参数	233
8.7 动态数组	237
8.8 集合类	240
8.8.1 队列.....	243
8.8.2 栈.....	245
8.8.3 有序表.....	247
8.8.4 哈希表.....	248
8.8.5 字典.....	250
8.8.6 Lookup 类	253

第 9 章 泛型 /256

9.1 使用泛型的原因	256
9.2 泛型简介	258
9.3 类型参数	261
9.3.1 类型参数命名规则.....	261
9.3.2 类型参数的约束.....	262
9.3.3 类型参数约束的特殊情况.....	265
9.3.4 类型参数默认值.....	266
9.4 泛型类	268
9.5 泛型接口	270
9.6 泛型方法	275
9.7 泛型委托	277
9.8 运行时中的泛型	278



9.9 泛型和数组	280
9.10 泛型和属性	281
9.11 C++ 模板和 C# 泛型的区别	282

第 10 章 反射 /283

10.1 泛型概述	283
10.2 反射中常用类	285
10.2.1 Assembly	285
10.2.2 Module	287
10.2.3 ConstructorInfo	288
10.2.4 MethodInfo	289
10.2.5 FieldInfo	290
10.2.6 EventInfo	292
10.2.7 PropertyInfo	293
10.2.8 ParameterInfo	294
10.2.9 CustomAttributeData	294
10.3 访问自定义属性	298
10.4 访问默认成员	300
10.5 使用反射将委托挂钩	300
10.6 反射的安全注意事项	305

第 11 章 字符串操作 /307

11.1 System.String 类	307
11.1.1 声明字符串变量	307
11.1.2 String 类的属性成员	309
11.1.3 字符串的查看和比较	310
11.1.4 分割字符串并获取子字符串	313
11.1.5 字符串的插入、删除和大小写转换	314
11.1.6 合并字符串	317
11.2 格式化字符串	319
11.2.1 格式化概述	320
11.2.2 标准数值格式	320
11.2.3 自定义数值格式	323
11.2.4 标准日期时间格式化	327
11.2.5 自定义日期时间格式化	330
11.2.6 枚举类型格式化	333
11.3 StringBuilder 类	335



第 12 章 文件与流 /338

12.1	System.IO 命名空间类	338
12.2	Path 类	339
12.3	用流读写文件	342
12.3.1	FileStream 类	342
12.3.2	BinaryReader 和 BinaryWriter 类	345
12.3.3	StreamReader 和 StreamWriter 类	350
12.3.4	序列化	352
12.4	File 类和 FileInfo 类	354
12.4.1	文件是否存在	355
12.4.2	创建文件	355
12.4.3	复制文件	357
12.4.4	移动文件	359
12.4.5	删除文件	360
12.4.6	获取和设置文件基本信息	360
12.5	Directory 类和 DirectoryInfo 类	362
12.5.1	目录是否存在	363
12.5.2	创建目录	365
12.5.3	删除目录	367
12.5.4	移动目录	367
12.5.5	遍历目录	371

第 13 章 多线程编程 /376

13.1	多线程的相关概念	376
13.2	Thread 类	377
13.2.1	线程的创建和启动	378
13.2.2	线程的挂起、恢复和终止	381
13.2.3	线程的状态	384
13.2.4	线程优先级	386
13.3	线程的同步	388
13.4	多线程的自动管理	395
13.4.1	线程池	395
13.4.2	计时器	398

第 14 章 Windows 窗体和控件 /401

14.1	Windows 窗体	401
14.1.1	创建 Windows 窗体	401



14.1.2	在项目中添加窗体	403
14.2	Windows 窗体控件	405
14.2.1	控件分类	405
14.2.2	控件的基本操作	407
14.2.3	命令控件	409
14.2.4	设置选项控件	412
14.2.5	列表选择控件	415
14.2.6	编辑文本控件	419
14.2.7	显示信息控件	422
14.2.8	日期选择控件	424
14.2.9	弹出式信息控件	427
14.2.10	图像控件	429
14.2.11	容器控件	430
14.3	对话框控件	432
14.3.1	ColorDialog 控件	432
14.3.2	FontDialog 控件	433
14.3.3	OpenFileDialog 控件	434
14.3.4	PrintDialog 控件	434
14.3.5	FolderBrowserDialog 控件	435
14.3.6	SaveFileDialog 控件	435
14.4	菜单和工具栏控件	436
14.4.1	MenuStrip 控件	436
14.4.2	ContextMenuMenuStrip 控件	437
14.4.3	ToolStrip 控件	438
14.4.4	ToolStripContainer 控件	438

第 15 章 数据库编程 /440

15.1	ADO.NET 概述	440
15.1.1	数据库与 ADO.NET	440
15.1.2	关于 ADO.NET 的类	443
15.1.3	Windows 应用程序与 ADO.NET	450
15.2	ADO.NET 应用	453
15.2.1	用 DataReader 从数据库中读取数据	453
15.2.2	用 DataSet 从数据库中读取数据	454
15.2.3	更新数据库的内容	457
15.2.4	访问数据集中的多个表	459
15.2.5	深入理解 ADO.NET 中的 SQL 语句	461
15.2.6	数据绑定	466



第 16 章 网络通信编程 /471

16.1	.NET Framework 中的请求和响应	471
16.2	TCP/IP 协议	472
16.2.1	IP 协议	473
16.2.2	TCP 协议	473
16.3	使用 TcpListener 和 TcpClient 收发信息	474
16.3.1	同步、异步、阻塞和非阻塞	474
16.3.2	使用 TcpListener 与 TcpClient	475
16.3.3	使用 Socket 类代替 TcpListener 和 TcpClient	479
16.4	典型的网络应用	483
16.4.1	下载网页	483
16.4.2	上传和下载文件	486
16.4.3	接收电子邮件信息	489
16.4.4	实现 ping 命令	494

第 17 章 异步编程 /504

17.1	异步编程概述	504
17.1.1	开始异步操作	504
17.1.2	结束异步操作	505
17.2	异步调用的常用方法	506
17.2.1	使用 EndOperationName 方法	506
17.2.2	使用 AsyncWaitHandle 对象	508
17.2.3	使用轮询	510
17.2.4	使用 AsyncCallback 委托	512
17.3	异步功能——Visual C# 2012 新特性	515
17.4	演练编写异步程序	517

参考文献 /528

第 1 章

Visual C# 2012 简介

微软公司从 20 世纪 90 年代开始就一直主导了桌面计算机的操作系统和开发环境,我们都是从视窗系统的一个个新版本、VS 环境的一个个新版本跟踪过来的,每个新版本总是有许多让老用户惊喜的改进和新特性,本书要介绍的 Visual C# 2012 亦不例外。如果你对某些内容已经熟悉,不妨向下翻页,直到看到让你惊喜的新特性。后面很可能用到前面介绍的某些知识,但既然书在手里,可以随时回头翻阅。

1.1 C# 概述

C# (读作 C Sharp) 是一种简单、现代、面向对象、类型安全的编程语言,开发人员可以使用它来构建在 .NET Framework 上运行的各种安全、可靠的应用程序。C# 源于 C、C++、Java,采三家之所长并增加了自己的特点。ECMA 和 ISO/IEC 都对 C# 做了标准化,分别为 ECMA-334 和 ISO/IEC23270,而 Microsoft Visual C# 2012 对这两个标准都提供了很好的支持。

1.1.1 C# 的产生与特点

在过去的二十多年里,C 和 C++ 已经成为商业软件开发领域中使用最为广泛的语言。它们为程序员提供了十分灵活的操作,不过却牺牲了一定的效率。与诸如 Microsoft Visual Basic 等语言相比,同等级别的 C/C++ 应用程序往往需要更长的时间来开发。同时,C/C++ 与其他一些语言相比又有其复杂性,所以程序员都试图寻找,是否有一种新的语言能在功能与效率之间达到一个更为理想的平衡点? C# 正是 Microsoft 对这一问题的解决方案。

1.1.1.1 C# 产生

C# 是在 2000 年由 Microsoft 公司 Anders Hejlsberg 和 Scott Wiltamuth 领导的小组开发的,作为 .NET 平台上的语言,使程序员可以方便地集成到 .NET。C# 主要沿袭了 C/C++ 语言,而语法又与 Java 相近。它摒弃了 C++ 复杂性,不再支持宏、模板和多重继承,使它更易用、更少出错。并且 C# 对 C++ 也提供了相应的扩充,如严格的类型安全、版本控制、垃圾收集(garbage collect)等。这些功能不仅提高了 C# 语言的安全性、可靠性,同时使 C# 更适合于开发组件级应用。

1.1.1.2 C# 特点

C# 作为一种针对 .NET 平台的开发语言,可以说是一个各种语言优点的集大成者:完全支持类和面向对象编程,包括接口和继承、虚函数和运算符重载的处理;定义完整、一致的基本类型集;对自动生成 XML 文档说明的内置支持;自动清理动态分配的内存;可以使用用户定义的特性来标记类和方法;对 .NET 基类库的完全访问权,并易于访问 Windows API;以 VB 的风格支持属性和事件;改变编译器选项,可以把程序编译为可执行文件或 .NET 组件库;可以用于编写 ASP.NET 动态 Web 页面和 XML Web 服务。

1.1.2 C# 4.0 新特性

回顾 C# 发展的历史,C# 1.0 模仿 Java,并保留了 C/C++ 的一些特性如 struct,新学者很容易上手;C# 2.0 加入了泛型,也与 Java 1.5 的泛型如出一辙;C# 3.0 加入了一些语法,并在没有修改 CLR 的情况下引入了 LINQ,简直是神来之笔,虽然很多项目出于各种各样如性能之类的原因没有采用,但非常适合小型程序的快速开发,减轻了程序员的工作量,也提高了代码的可读性;C# 4.0 增加了动态语言的特性,从里面可以看到很多 JavaScript、Python 这些动态语言的影子。虽然越来越偏离静态语言的道路,但从另一个角度来说,这些特性也都是为了提高程序员的生产力。

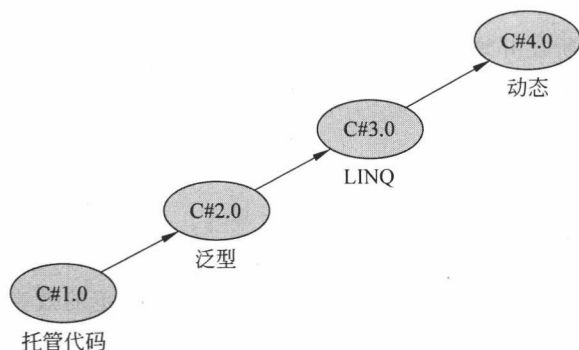


图 1.1 C# 的演化

C# 4.0 的主题就是动态编程(Dynamic Programming)。虽然 C# 仍然是一种静态语言,但是对象的意义开始变得越来越“动态”。它们的结构和行为无法通过静态类型来捕获,或者至少编译器在编译程序时无法得知对象的结构和行为。C# 4.0 包括大量的增强和新增特性,其主要新特性如下所示。

1. 支持动态查找

动态查找允许在编写方法、运算符和索引器调用、属性和字段访问甚至对象调用时,绕过 C# 静态类型检查,而在运行时进行解析。

2. 同时支持参数命名和可选参数

现在 C# 中的参数可以通过在成员声明中为其提供默认值来指名它是可选的。在调用该成员时,可选参数可以忽略。另外,在传入任何参数时都可以按照参数名而不是位置

进行传递。

3. 增强的 COM 互操作特性

动态查找以及命名参数和可选参数都有助于使针对 COM 的编程不再像现在这样痛苦。在这些特性之上,增加了大量其他小特性,进一步改善了互操作体验。

4. 变性

C# 4.0 中,把过去的 `IEnumerable<string>` 变为现在的 `IEnumerable<object>`,并包含了类型安全的“协变性和逆变性”(co-and contravariance),而且通用的 BCL 也将利用这一特性进行更新。

1.1.2.1 动态查找

动态查找可以用统一的方式来动态调用成员。有了动态查找,当你拿到一个对象时,不用管它是来自于 COM 还是 IronPython、HTML DOM 或是反射,只需要对其进行操作即可。运行时帮你指出针对特定的对象,以及这些操作的具体意义。这种方法带来了巨大的灵活性,并能极大程度地精简代码,但它伴随着一个巨大的缺点——不会为这些操作维护静态类型。在编译时,会假设动态对象支持任何操作,而如果它不支持某个操作,则只有到运行时才会出现错误。有的时候这不会有任何损失,因为对象根本不具有静态类型,而且其他情况下必须在简洁和安全之间进行权衡。为了帮助进行权衡,C# 的一个设计目标就是允许在每个单独的调用中选择是否使用动态行为。

1. 动态类型

C# 4.0 引入了一个新的静态类型,称为 `dynamic`。当你拥有了一个 `dynamic` 类型的对象后,你“对它做的事情”只会在运行时进行解析。

```
dynamic d=GetDynamicObject(...);  
d.M(7);
```

C# 编译器允许你使用任何参数在 `d` 上调用一个方法,因为它的类型是 `dynamic`。运行时检查 `d` 的实际类型。可以认为 `dynamic` 类型是 `object` 类型的一个特殊版本,指出了对象可以动态地使用。选择是否使用动态行为很简单——任何对象都可以隐式转换为 `dynamic`,“挂起信任”直到运行时。反之,从 `dynamic` 到任何其他类型都存在“赋值转换”,可以类似于赋值的结构中进行隐式转换。

```
dynamic d=7; //隐式变换  
int i=d; //赋值转换
```

2. 动态操作

不仅是方法调用,字段和属性访问、索引器和运算符调用甚至委托调用都可以动态地分派。

```
dynamic d=GetDynamicObject(...);  
d.M(7); //调用方法  
d.f=d.P; //获取并设置字段属性
```

```
d["one"]=d["two"];           //通过索引进行获取和设置
int i=d+3;                     //调用运算符
string s=d(5,7);              //调用委托
```

C# 编译器在这里的角色就是打包有关“在 d 上做什么”的必要信息,使得运行时可以获取这些信息,并检测对于实际对象 d 这些操作的确切含义。可以认为这是将编译器的部分工作延迟到了运行时。任何动态操作的结果本身也是 dynamic 类型的。

3. 运行时查找

如果 d 是一个 COM 对象,则操作通过 COM IDispatch 进行动态分派。这允许调用没有主互操作程序集(Primary Interop Assembly,PIA)的 COM 类型,并依赖 C# 中没有对应概念的 COM 特性,如索引属性和默认属性。

如果 d 实现了 IDynamicObject 接口,则请求 d 自身来执行该操作。因此通过实现 IDynamicObject 接口,类型可以完全重新定义动态操作的意义。这在动态语言,如 IronPython 和 IronRuby 中有大量使用,用于实现它们的动态对象模型。API 也会使用这类对象,例如 HTML DOM 允许直接使用属性语法来访问对象的属性。

除此之外,如果 d 是一个标准的 .NET 对象,操作是通过在其类型上进行反射来分派的,C# 的“运行绑定器”(runtime binder)实现了运行时的 C# 查找和重载解析。其背后的本质是将 C# 编译器作为运行时组件运行,来“完成”被静态编译器延迟的动态操作。

1.1.2.2 命名参数和可选参数

命名参数和可选参数是两个截然不同的功能,但通常一起使用。在进行成员调用时,可以忽略可选参数;而命名参数的方式可以通过名称来提供一个参数,而无须依赖它在参数列表中出现的位置。有些 API,尤其是 COM 接口(如 Office API)本身就是通过命名参数和可选参数编写的。即便是编写 .NET 中的 API,你也会发现很多时候你在被迫为不同的参数组合方式编写一个方法的大量重载形式,以便给调用者提供最高的可用性。在这种情况下,可选参数就会成为一种非常有用的替代方式。

1. 可选参数

为一个参数提供默认值就可以将其声明为可选的。

```
public void M(int x, int y=5, int z=7);
```

这里的 y 和 z 就是可选参数,在调用时可以忽略。

```
M(1, 2, 3);           //正常调用 M
M(1, 2);               //省略 z,相当于 M(1,2,7)
M(1);                  //省略 y 和 z,相当于 M(1,5,7)
```

2. 命名的和可选的实参

C# 4.0 不允许忽略逗号之间的实参,比如 M(1,,3)。否则会导致大量不可读的、需要“数逗号”的代码。替代方式是任何参数都可以通过名字传递。因此,如果在调用 M 时