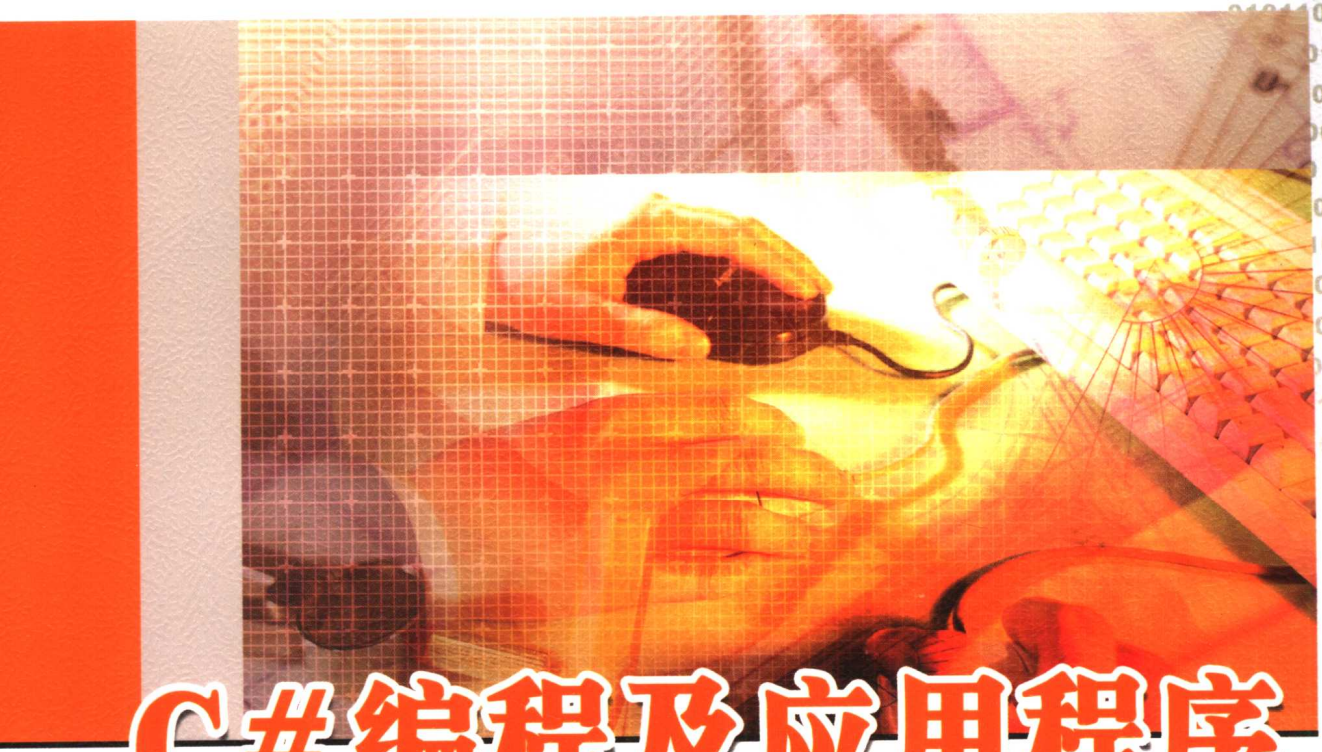
高等院校计算机教育系列教材



C#编程及应用程序开发教程（第2版）

刘 烨 季石磊 等编著

- 阅读本书无需编程经验
- 可直接在Visual Studio.NET 开发环境中学习C#语言
- 通过实例应用解析语义规范
- 内容简明扼要，突出知识要点
- 提供丰富的相关背景知识
- 追求现代教育理念

赠送电子课件

清华大学出版社

高等院校计算机教育系列教材

C#编程及应用程序开发教程
(第2版)

刘　烨　季石磊　等编著

清华大学出版社
北　京

内 容 简 介

C#语言是 Microsoft 公司为推行.NET 战略而发布的一种全新的、彻底的、面向对象的编程语言，它融C++的强大功能和 Visual Basic 的简易性于一体，具有清晰的面向对象的语法结构、优秀的编程开发环境和高效率的编译工具。

本书从结构上分为两个部分。其中 1~16 章为 C#语言程序设计基础，将 C#语言的各种语法知识点按循序渐进的方式编排，并提供了丰富的示例。17～21 章介绍了在.NET 平台上如何使用 C#语言开发各种应用程序，包括：创建 Windows 应用程序、C#组件编程、C#数据库编程、Web 应用程序、.NET 报表设计以及.NET 软件发布等，帮助读者在.NET 平台上开发、测试和发布各种应用程序。

阅读本书的读者无需编程经验，可以是在校学习的各专业的研究生、本科生或大专生，或企、事业单位的初、中级用户。本书还可作为广大计算机初、中级爱好者的教材或参考书。

图书在版编目(CIP)数据

C#编程及应用程序开发教程(第 2 版)/刘烨，季石磊等编著. —北京：清华大学出版社，2007.5
(高等院校计算机教育系列教材)
ISBN 978-7-302-14874-6

Ⅰ.C… Ⅱ.①刘… ③季… Ⅲ.C 语言—程序设计—高等学校—教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2007)第 036724 号

责任编辑：彭　欣
封面设计：山鹰工作室
版式设计：杨玉兰
责任校对：周剑云
责任印制：李红英
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 刷 者：北京市清华园胶印厂
装 订 者：三河市李旗庄少明装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：32　**字　数**：774 千字
版　　次：2007 年 5 月第 2 版　　**印　　次**：2007 年 5 月第1 次印刷
印　　数：1～4000
定　　价：43.00 元

前　言

随着网络计算时代的到来，各种应用于网络服务的计算机语言、操作系统和开发工具应运而生，C#语言就是 Microsoft 公司为推行.NET 战略而发布的一种全新的、彻底的、面向对象的编程语言，它具有清晰的面向对象的语法结构、优秀的编程开发环境和高效率的编译、测试和发布工具。在 Visual Studio .NET 框架下使用 C#语言，不仅可以编写 Windows 图形界面程序和数据库应用程序，还可以进行构件编程、多线程编程和分布式编程等。

本书第 1 版出版以来，受到广大读者的欢迎。为了满足教学和软件开发实践的进一步需求，同第 1 版相比，本书增加了第 14 章第 5 节.NET 环境下的单元测试；第 21 章.NET 报表设计；其他各章也相应作了适当修改及调整。

新版中保留了以下特色：

- 阅读本书无需编程经验。读者如果有 C、C++和 Java 编程经验，阅读本书后便可立即进入实战状态。
- 本书让读者直接在 Visual Studio .NET 开发环境中学习 C#语言。这样可以使读者熟悉 .NET 开发环境，为今后深入学习此环境下的各种高级编程技术打下基础。
- 本书仅是学习 C#编程的一个起点。C#编程不只是包含本书介绍的语法和定义，还包括 .NET 类库及其他高级编程技术，学习完本书后读者一定具有自学 C#的其他编程技术的能力了。

本书的内容编排有以下特色：第 1 版中包含的许多辅助教学和实践的结构和内容编排特色，深受广大学生的欢迎，在这一版仍继续保留。本书采用讲授计算机语言的常规做法，将各知识点按循序渐进的方式编排，力求文字通俗易懂，脉络清晰。为便于读者理解和掌握新概念和新问题，特意在各章节的相关知识点后用“小资料”、“说明”和“提示”来帮助读者领会实质。

- 小资料：提示与所介绍知识点相关的背景知识，或其他语言对该知识点的处理方式与 C#语言的区别。
- 说明：对介绍的知识点的进一步理解。
- 提示：应用中需注意的问题。

各章节内容安排如下：

第 1 章介绍 Microsoft .NET 平台。向读者简要介绍 .NET 平台结构及环境。

第 2 章介绍 Visual Studio .NET 集成开发环境，并通过一个简单的 C#语言编写的程序，分析 C#程序的构成要素和程序结构特征。

第 3～11 章介绍 C#语言的基本语法知识和使用方法，同时给出相应知识点的应用示例，以便读者领会 C#语言的实质。

第 12～16 章介绍 C#的一些高级编程技术和程序的调试及测试方法，以便扩展用户的编程思路同时满足其进行高级编程的需求。

第 17～21 章帮助读者在.NET 平台上使用 C#语言开发各种应用程序。

本书对绝大多数的知识点都给出了完整的程序代码，并对程序运行过程中可能出现的现象和结果给予了较详细的解释说明，以使初学者不至于无从下手。所以本书非常适合广

大程序设计的初学者，尤其适合在校的广大计算机应用编程爱好者学习。本书也可作为初、中级程序员“品味”C#语言新特性的一个有效途径。

本书第 1、7、8、9、10、11、19 章由刘烨编写；第 4、12、13、14、15 章由马明编写；第 5、6、16、20 章由李莹编写；第 2、3、21 章由季石磊编写；第 17 章由魏维编写；第 18 章由张宇宁编写，参加本书编写工作的还有：李万勇、李占稳、王玲风、李成丽等。

在本书成稿后，再次邀请天津大学精密学院李刚教授和林凌教授对本书进行了审核，他们对全书给予了认真的修改和审定，并提出了宝贵的修改建议，在此表示衷心的感谢！

由于水平有限，书中难免有不妥之处，敬请读者批评指正。

作 者

目　录

第 1 章　Microsoft .NET 平台 1

1.1　网络计算时代 1
1.2　Microsoft .NET 平台 2
1.3　.NET Framework 4
1.3.1　公共语言运行时 4
1.3.2　基础类库 9
1.3.3　ADO.NET 和 XML 10
1.3.4　基于 ASP.NET 编程框架上的 WebForms 和 Web Service 11
1.3.5　WinForms 用户界面 13
1.3.6　.NET Framework 的优势 13
1.4　新一代编程语言 C# 13
1.4.1　C#的新特性 14
1.4.2　C#与 C++ 18
1.4.3　C#与 Visual Basic .NET 19
1.4.4　C#与 Java 19

第 2 章　C#编程和编译环境 23

2.1　Visual Studio .NET 集成开发环境 23
2.1.1　.NET 集成开发环境(IDE)简介 24
2.1.2　创建项目 27
2.1.3　编写代码环境 29
2.2　一个简单的 C#程序 32
2.3　编辑、编译和运行一个 C#程序 38

第 3 章　数据类型和变量 41

3.1　变量和常量 42
3.1.1　变量 42
3.1.2　常量 49
3.2　数值类型 49
3.2.1　整数类型 52
3.2.2　字符类型 53
3.2.3　浮点数类型 56
3.2.4　小数类型 57
3.2.5　布尔型 58
3.2.6　检查和不检查 60
3.3　引用类型 60
3.3.1　对象类型 62
3.3.2　字符串类型 62
3.4　各种简单类型的数据间转换 64
3.5　装箱和拆箱转换 67
3.5.1　装箱转换 67
3.5.2　拆箱转换 68

第 4 章　运算符和表达式 71

4.1　C#运算符概述 71
4.2　赋值运算符及其表达式 73
4.3　算术运算符及其表达式 74
4.3.1　加法和减法运算符 74
4.3.2　自增和自减运算符 75
4.3.3　乘法和除法运算符 76
4.3.4　取余运算符 77
4.4　关系运算符及其表达式 78
4.5　逻辑运算及其表达式 79
4.6　位运算符及其表达式 80
4.7　条件运算符及其表达式 83
4.8　其他运算符 83
4.8.1　new 运算符 83
4.8.2　sizeof 运算符 83
4.8.3　is、as 和 typeof 运算符 84
4.8.4　checked 和 unchecked 运算符 86

第 5 章　数据的输入和输出 88

5.1　控制台输入 88
5.1.1　Console.Read()方法 88
5.1.2　Console.ReadLine()方法 89
5.2　控制台输出 90
5.2.1　数据的格式化 90
5.2.2　格式化说明符 93

5.3 处理字符串的方法 101
5.3.1 String 类的字符串方法 101
5.3.2 StringBuilder 类的字符串方法 102
5.3.3 Parse( )方法 104
5.3.4 Convert 类 104

第 6 章 程序控制语句 107

6.1 选择结构程序设计 107
6.1.1 if 语句 107
6.1.2 switch 语句 110
6.1.3 程序举例 113
6.2 循环结构程序设计 115
6.2.1 while 语句 115
6.2.2 do-while 语句 116
6.2.3 for 语句 117
6.2.4 foreach-in 语句 119
6.2.5 程序举例 126
6.3 break 语句、continue 语句和 goto 语句 127
6.3.1 break 语句 127
6.3.2 continue 语句 128
6.3.3 goto 语句 129
6.3.4 程序举例 129

第 7 章 数组 133

7.1 一维数组 133
7.1.1 一维数组的定义和初始化 133
7.1.2 引用一维数组元素 135
7.2 二维数组 139
7.2.1 二维数组的定义和初始化 139
7.2.2 初始化二维数组 140
7.2.3 引用二维数组元素 141
7.3 不规则数组 146
7.3.1 二维不规则数组的定义和初始化 147
7.3.2 二维不规则数组的引用 148
7.4 使用 System.Array 类的方法和属性 150
7.4.1 Array 类的属性 150
7.4.2 使用 Array 类构造数组 152
7.4.3 使用 Array 类的方法 153

第 8 章 类 159

8.1 面向对象程序设计思想 159
8.2 类及其构成 162
8.3 创建对象 164
8.4 类的数据成员 165
8.4.1 常量成员 165
8.4.2 变量成员 166
8.4.3 将类定义为数据成员 176
8.5 类的方法成员 178
8.5.1 定义类的方法成员 178
8.5.2 使用方法 180
8.5.3 带参数的方法 184
8.5.4 静态方法 190
8.5.5 方法重载 191
8.5.6 其他类方法 194
8.5.7 递归方法 202
8.6 类的属性成员 203
8.7 索引器 208
8.8 运算符重载 210
8.8.1 重载运算符方法 210
8.8.2 重载双目运算符 211
8.8.3 重载单目数学运算符 215
8.8.4 重载关系运算符 216

第 9 章 继承与多态 218

9.1 继承机制 218
9.1.1 继承的概念 218
9.1.2 继承的工作机制 220
9.1.3 派生类的构造函数和析构函数 223
9.1.4 base 关键字的另一用途 225
9.1.5 隐藏基类成员 225
9.1.6 使用 protected 保护访问方式 227
9.1.7 使用 internal 内部访问方式 227
9.2 多态性和虚方法 229

9.2.1 多态性 229
9.2.2 虚方法 232
9.3 抽象类和抽象方法 237
9.3.1 抽象类 237
9.3.2 抽象方法 238
9.4 密封类和密封方法 244
9.4.1 密封类 244
9.4.2 密封方法 245
9.5 终极基类 Object 246
9.5.1 Object 类中的方法 246
9.5.2 装箱/拆箱前后的数据类型 248
9.6 类转换 249
9.6.1 关键字 is 249
9.6.2 关键字 as 250
9.6.3 不同类型的对象组成的数组 251

第 10 章 接口、代理和事件 254

10.1 接口 254
10.1.1 接口与类 254
10.1.2 接口的定义 255
10.1.3 接口的实现与使用 260
10.1.4 接口映射 264
10.1.5 显式接口成员实现 265
10.1.6 接口实现的继承 269
10.1.7 接口的重新实现 274
10.1.8 接口的查询 276
10.2 代理 277
10.2.1 代理的概念 277
10.2.2 代理的定义 278
10.2.3 代理的使用 279
10.2.4 Delegate 类和 MulticastDelegate 类 282
10.2.5 多重代理的实现 286
10.3 事件 288
10.3.1 事件的概念 288
10.3.2 创建事件和使用事件 288
10.3.3 多播事件 297

第 11 章 结构和枚举 300

11.1 结构 300
11.1.1 结构与类 300
11.1.2 结构与结构成员的定义 301
11.1.3 使用和访问结构成员 302
11.1.4 结构的嵌套 304
11.1.5 结构与类的区别 305
11.2 枚举 306
11.2.1 枚举的定义 306
11.2.2 使用和访问枚举成员 309

第 12 章 命名空间 311

12.1 命名空间 311
12.2 定义命名空间 311
12.3 使用命名空间 312
12.3.1 命名空间的完全限定名 312
12.3.2 C#应用程序的组织结构 314
12.3.3 using 语句 314

第 13 章 异常处理 320

13.1 异常处理的概念 320
13.2 C#的异常控制机制 322
13.2.1 使用 try/catch 语句抛出和处理异常 322
13.2.2 使用 throw 语句抛出异常 327
13.2.3 使用 finally 语句添加最后执行的操作 329
13.3 .NET Framework 中的异常类 331
13.3.1 C#异常处理的内部机制 331
13.3.2 .NET Framework 中的异常类 332
13.4 自定义异常类 336

第 14 章 编译预处理和调试/测试技术 339

14.1 程序中的错误 339
14.2 逐行检查代码 340
14.3 编译预处理 340
14.3.1 定义预处理 340
14.3.2 条件编译预处理 341
14.3.3 输出代码中的错误和警告 344
14.3.4 修改行号 345

14.3.5 区域预编译 347
14.4 使用调试工具 348
14.4.1 几个基本调试概念 348
14.4.2 常用的调试策略 350
14.4.3 程序的调试信息 351
14.5 .NET 环境下的单元测试.....355
14.5.1 NUnit 单元测试工具 356
14.5.2 使用 NUnit 测试工具 357

第 15 章 代码属性 362

15.1 概述 362
15.2 使用代码属性 362
15.3 .NET 框架下的预定义属性类.....365
15.3.1 带参数的属性 365
15.3.2 System.Attribute 类 365
15.3.3 描述程序集和模块的属性 ...366
15.3.4 描述源代码的代码属性 368
15.4 自定义代码属性类 373
15.5 检索有关的代码属性信息 376

第 16 章 不安全代码.....381

16.1 不安全代码 381
16.2 不安全的代码块 381
16.2.1 指针 382
16.2.2 unsafe 关键字.....383
16.2.3 fixed 语句.....384
16.3 C#程序中的指针.....386
16.3.1 用指针访问对象 386
16.3.2 指针运算 388

第 17 章 创建 Windows 应用程序 390

17.1 什么是 Windows 窗体.....390
17.2 创建简单的 WinForm 程序.....390
17.3 Windows 窗体应用程序模型 393
17.3.1 窗体 394
17.3.2 属性 394
17.3.3 控件 397
17.3.4 事件 400
17.3.5 Windows Forms 程序设计的步骤 401
17.4 WinForm 控件.....401
17.4.1 常用控件 401
17.4.2 示例 401
17.5 Visual C#的菜单设计与编程.....409
17.5.1 菜单设计基础知识.....409
17.5.2 示例 410
17.5.3 用程序完成菜单设计.....412
17.6 Visual C#中的 MDI 编程.....417

第 18 章 C#组件编程.....421

18.1 用 C#做类库.....421
18.1.1 制作一个组件 421
18.1.2 使用 DLL.....425
18.2 用 C#做自定义控件.....427
18.2.1 创建控件 427
18.2.2 使用控件 431
18.3 用 C#做用户控件.....431
18.3.1 控件制作 431
18.3.2 使用组件 434

第 19 章 C#数据库编程.....438

19.1 ADO.NET 数据库访问 438
19.1.1 ADO.NET 概述 438
19.1.2 ADO.NET 的数据访问对象 440
19.2 ADO.NET 访问常用数据库 450
19.2.1 SQL Server 450
19.2.2 Oracle.....451
19.2.3 Access.....452
19.3 C#数据库的 Windows 编程.....452
19.4 C#数据库的 Web 编程 458

第 20 章 开发 Web 应用程序 461

20.1 ASP.NET 的开发环境配置.....461
20.2 编写 ASP.NET Web 应用程序 462
20.3 ASP.NET 服务器端控件 463
20.3.1 Web 服务器控件 463
20.3.2 HTML 服务器控件 467
20.3.3 验证控件 468
20.3.4 用户控件 471

20.4 创建 Web 服务.....474
20.4.1 Web 服务.....474
20.4.2 一个简单的 Web 服务.....474
20.4.3 使用 Web 服务访问数据库.....476
第 21 章 .NET 报表设计.....480
21.1 OWC 组件.....480
21.1.1 OWC 组件概述.....480
21.1.2 OWC 组件的内部控件.....481
21.1.3 OWC 组件应用示例.....483
21.1.4 Spreadsheet 控件使用语法参考.....487
21.2 SQL Server Reporting Services.....490
21.2.1 报表制作环境.....490
21.2.2 创建简单的报表.....491
21.3 Crystal Reports.....494
21.3.1 Crystal Reports 概述.....494
21.3.2 报表设计.....495
21.3.3 创建简单报表.....497
21.3.4 用 CrystalReportViewer 报表查看器承载报表.....499

第 1 章　Microsoft .NET 平台

Microsoft .NET 开发平台的发布，标志着近 10 年来 Microsoft 公司开发平台一个重大的转变。这个开发平台包括一个用于加载和运行应用程序的新的软件基础结构(.NET Framework 和 ASP.NET)，以及支持该结构的最佳编程语言 C#。使用 Microsoft 公司开发技术的开发者一直习惯了使用 ASP 进行 Web 编程，使用 Visual Basic 和 Visual C++进行 Win32 编程，基于 COM、DCOM 技术设计应用程序，那么他们如何利用 .NET 的开发技术和工具构建下一代的互联网应用呢？

本章学习重点：

- 了解 Microsoft .NET 平台
- 了解 .NET Framework 的构成和特点
- 了解新一代面向对象语言 C#的优势
- C#与其他语言(C++、Visual Basic .NET、Java)

1.1　网络计算时代

对于计算机工业，革命就是一种“生活方式”。从大型计算机时代，到个人计算机和多媒体技术的出现，Internet 的接踵而至，这些都革命性地改变了我们的生活方式、工作手段和思维理念。

信息技术的高速发展推动了计算模式不断更新，计算模式的改变导致了计算机应用软件开发观念的根本改变。在近 20 多年里，计算模式经历了以下阶段。

- 单主机时代的主机/终端计算模式(MainFram Computing)
- 文件服务器时代的共享数据模式(File Server Computing)
- 分布计算时代的客户/服务器计算模式(Distributed Client/Server Computing，简称 C/S)
- 电子商务时代的浏览/服务器网络计算模式(Browse/Server Network Computing，简称 B/S)

我们现在正进入网络计算时代，未来的世界是一个数字世界。网络计算模式的软件开发理念是：软件就是服务。在网络计算时代，将会有许多应用服务开发商提供各种性能良好的构件(软件 IC 块)放在自己的服务器上，供网上用户在任何设备、任何时间和任何地点使用。用户不需掌握某种计算机程序设计语言，只需全力运用本行业知识设计计算模型，然后向开发商订购解决方案(数字定制)。用户甚至不用下载，只要按时交纳使用(或服务)费，即可用到开发商提供的解决方案，按方案备齐软、硬件设备，组装好服务构件即可实现自己的应用系统。另外，用户也不用操心买来构件的升级版本，出了问题，打个电话，开发商或供应商就会上门诊断，进行维护等售后服务。这种所订购的构件就相当于是一个机器零件(配件)或一个完整的产品，用户只需买回组装或全盘使用，即可集成出一个应用系统。在这个时代，80%~90%的应用都由软件开发商提供，这将极大地推动各行各业的信

息化、自动化的发展。

新一代网络计算模式的技术特征是：HTTP+WWW+Java，由这 3 项技术支持的网络计算具备两个要素：

- 开发的软构件随处可移，是平台无关的，以便各种构件能很好地协作集成。
- 有一个统一表达、描述信息的标准协议，解决了异构系统间的通信问题，实现真正的网络计算。

Microsoft 公司于 2000 年 6 月在专业开发者大会上，发布了下一代网络计划(NGWS)——Microsoft .NET(以下简称 .NET)，这为开发商进行网络计算应用提供了两个强有力的技术支持：一是 .NET 平台提供了 .NET 的虚拟机，所有的程序都事先从源代码被编译成与处理器无关的中间语言(Microsoft Intermediate Language，MSIL)，只当程序运行时，才在即时编译器(Just-In-Time，简称 JIT)的编译下，由中间语言代码编译成针对特定 CPU 类型和操作系统的本机机器代码运行，这实现了程序跨平台可移植性；二是本书要介绍的 C#(发音：C sharp)编程语言，C#是一种全新的、彻底的面向对象的语言，它结合了 C/C++和 Visual C++的强大功能以及 Visual Basic 的易用性。另外，无论在语法方面，还是在丰富的 Web 开发支持及自动化的内存管理上，C#都和 Java 非常相似。

.NET 平台最推崇的是用于开发新型的应用程序：Web Service，或者和 Web 上其他应用程序交换 XML 格式数据的服务器应用程序。.NET 使用 XML 来描述如 DCOM、CORBA 等远程过程调用，并通过简单对象访问协议(SOAP)提供的标准远程过程调用(PRC)方法来调用 Web Service，从而实现了在分布式应用构架内集成或协调 Internet 上的任意资源，即任何平台上任何应用都可以直接调用网络服务，由此解决了异构系统间的通信问题，实现了真正的网络计算。

1.2 Microsoft .NET 平台

.NET 平台为创建新一代分布式 Web 应用提供了所有工具和技术(表示技术、构件技术和数据库技术)。.NET 平台支持标准的 Internet 协议，包括 HTTP(超文本传输协议)、XML(可扩展标记语言)和 SOAP(简单对象访问协议)，从而实现了异构系统间应用程序的集成和通信，即用户和供应商可将在此平台上开发的产品和服务(开发商提供)无缝地嵌入自身的业务进程和日常活动的电子架构中。

.NET 平台由 5 个主要部分组成，如图 1.1 所示。

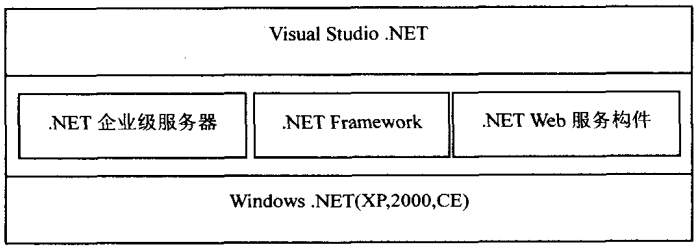


图 1.1 .NET 平台概貌

- Windows .NET
- .NET 企业级服务器(.NET Enterprise Servers)

- .NET Web 服务构件
- .NET Framework
- Visual Studio .NET

1. Windows .NET

Windows .NET 是可以运行.NET 程序的操作系统的统称，主要包括 Windows XP/2000/2003/CE 以及即将发布的 Windows Vista 等操作系统。还提供各种应用软件服务，包括：

- 因特网信息服务系统(IIS)
- 活动服务页面(ASP+)
- 活动目录服务(Active Directory)
- Microsoft 公司报文队列服务(MSMQ)
- Actiuex 数据对象+对象链接数据库服务(ADO+OLEDB)
- 构件对象模型+Microsoft 公司交易服务系统(COM+/MTS)
- 分布式事务处理协调器(Distributed Transaction Coordinator)

2. .NET Enterprise Servers

.NET Enterprise Servers 称为 Microsoft .NET 企业级服务器，这是 Microsoft 公司推出的进行企业集成和管理所有基于 Web 的各种服务器应用的系列产品，包括 Application Center 2000、BizTalk Server 2000、Commerce Srever 2000、SQL Server 2000、Exchange Server 2000 等。它的主要目标是帮助企业快速集成并架构电子商务及其应用系统。

3. .NET Framework

.NET Framework 是 .NET 的核心部分，它提供了建立和运行 .NET 应用程序所需要的编辑、编译等核心服务。

.NET Framework 主要由两部分组成：一是公共语言运行时(Common Language Runtime，CLR)环境，CLR 提供了一个可靠而完善的多语言运行环境，简化了应用程序的开发配置和管理，从而实现组件能在多语言环境下、跨平台工作；二是 .NET 的基础类库(Basic Class Library，BCL)，它提供了几乎所有应用程序都需要的公共代码。使用 .NET 类库提供的公共方法开发应用程序，可以使开发者将精力集中于编写应用程序所独有的代码，而不必重复编写类似读写文件的经常使用的功能代码。

4. Visual Studio .NET

Visual Studio .NET 是为建立基于 .NET Framework 应用程序而设的一个可视化集成开发环境(Integrated Development Enviroment，IDE)。它为所有的编程语言提供了简单统一的代码编辑器，包括 XML 编辑器、HTML 编辑器、SQL Server 接口、以图形化的方法设计服务器端构件的设计器、监控远程机器的 Server Explorer。可以说，Visual Studio .NET 集中了建立分布式应用所需的功能。

5. .NET Web 服务构件

要保证 .NET 能够正常运行，需要提供一些公用性 Web 服务，包括身份认证、发送信

息、个性化服务、系统化存储、日历、目录、搜索和软件传输等。.NET Web 服务构件就提供了一系列高度分布、可编程的公用性网络服务。.NET Web服务构件可以从任何支持SOAP的平台上访问，这些服务可以运行在公司内部的服务器上，被内部局域网的机器访问，也可以以 Internet 的方式发布和访问。

1.3 .NET Framework

.NET Framework 实际上是运行在 Windows 系列操作系统上的一个系统应用程序。它采用一种全新的网络计算机模式，通过标准的 Internet 协议如 XML 和 SOAP 等，解决了异构平台上的分布式松耦合计算问题。

.NET Framework 提供了一个语言无关的 CLR 来管理各种代码的执行过程，并为所有 .NET 语言开发各种 Web 应用和服务提供了框架公用类库的 FCL(Framework Class Library)，FCL 包括基础类库(BCL)和用户接口库。

.NET Framework 包括以下组件(如图 1.2 所示)：

- 公共语言运行环境(CLR)
- 基础类库(BCL)
- 数据库访问组件(ADO.NET 和 XML)
- 基于 ASP.NET 编程框架的网络服务(Web Service)和网络表单(WebForms)
- Windows 桌面应用界面编程组件(WinForm)

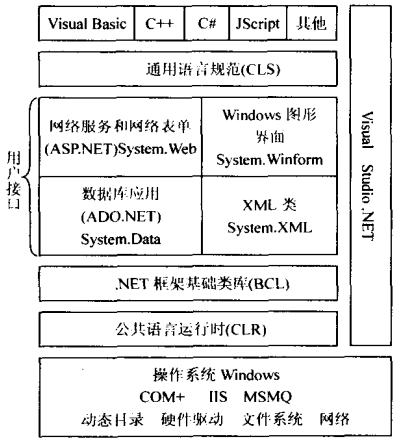


图 1.2 .NET Framework

.NET 整个开发框架是一组用于建立 Web 服务器应用程序和 Windows 桌面应用程序的软件组件，用该平台创建的应用程序是在 CLR(底层)的控制下运行的；在开发技术方面，.NET 提供了全新的数据库访问技术 ADO.NET、网络应用开发技术 ASP.NET 和 Windows 编程技术 WinForm；在开发语言方面，.NET 提供了 Visual Basic、Visual C++、C#、JScript 等多种语言支持，而 Visual Studio .NET 则是全面支持 .NET 的开发工具。

1.3.1 公共语言运行时

.NET Framework 的最底层是 CLR，它是 .NET Framework 的核心，管理着 .NET 代码

的执行。CLR 是一个软件引擎，用于加载应用程序、检查错误、进行安全许可认证、执行和清空内存。

运行时(Runtime Environment，简称 Runtime)是指那些支持在特定的平台上，用于运行特定编程语言编写的软件的库和程序集，它一般要处理软件和操作系统之间的接口细节，例如，系统调用、程序的启动和终止、内存管理等。例如：要运行 Visual Basic 6，则该计算机上必须存在 MSVisual BasicVM60.DLL 文件；同样，要运行 Java 程序，则目标计算机上必须存在 JVM(Java 虚拟机)。运行时分为 3 种：纯静态环境(如 Fortran)、基于堆栈环境(如 C、C++、Pascal)和纯动态环境(如 SmallTalk、Java)。CLR 是属于纯动态运行时的一种，它的主要组成部分是虚拟执行引擎 VEE(Virtual Execution Enging)。传统的“运行时”观念使程序员必须了解应用程序所运行的目标平台，从而导致一些语言和平台的限制。.NET 下的 CLR 的设计目的就是要打破这种限制。

1. CLR 的构成

如图 1.3 所示，CLR 由以下 11 个功能器件组成。

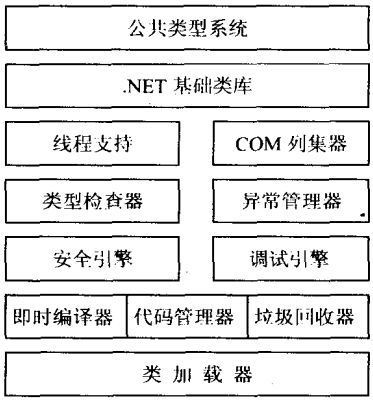


图 1.3 CLR 的构成

- 类加载器(Class Loader)：将应用程序的汇编加载到内存中。汇编包括微软中间语言(Microsoft Intermediate Language，MSIL)代码、描述应用程序中组件的元数据(类和类的布局描述)，以及其他应用程序所需的组件。
- 即时编译器(Just-In-Time，JIT)：负责将 MSIL 翻译成本地执行代码。
- 代码管理器(Code Manager)：管理代码的执行。
- 垃圾回收器(Garbage Collection)：负责整个 .NET 运行时托管代码的内存分配与释放任务，它通过一定的优化算法选择收集对象和时间，并进行自动的垃圾收集。
- 安全引擎(Security Engine)：提供基于认证的安全机制，如用户身份。
- 调试引擎(Debugger)：使开发者能调试和跟踪应用程序代码。
- 类型检查器(Type Checker)：检查并禁止非安全的类型转换及未初始化的变量的使用。
- 异常管理器(Exception Manager)：提供结构化的异常处理，与 Windows 结构化异常处理机制(SHE)集成，改进了错误报告。
- 线程支持(Thread Support)：提供了多线程编程的类和接口。

- COM 列集器(COM Marshaler)：处理与 COM 之间的配置。
- .NET 基础类库(BCL)：集成具有支持 .NET Framework 类库运行时的代码。

2. 公共类型系统(CTS)

为了实现语言的互用性，.NET Framework 采用以下的两种方法来解决语言的划分问题。

- 标准化数据类型。建立通用语言运行环境中的公共类型系统(Common Type System，CTS)，它为最常用的数据类型(如整数、实数、文本字符)定义了标准的内部描述和运算，并提供了将这些类型向所有的 .NET 语言和 CLR 扩展的机制。这种机制能够表示绝大多数现代编程语言的语法，消除了每种语言自己唯一且不兼容的方法。CTS 是一套 CLR 中的数据类型都必须遵守的规则。如果某种语言在创建数据类型时遵守了 CTS，则它创建和存储的数据将能够与其他也遵守了 CTS 的编程语言　兼容。
- 标准化应用程序格式。 .NET 拥有自己的(微软中间语言)MSIL、元数据和清单的汇编。所有 .NET 语言的编译器都生成这种格式。即通过从元数据中提取有关的 MSIL 的信息，编译器、调试器和协调器等工具可以分析处理任何一种源程序设计语言的数据。

3. CLR 工作原理

一个应用程序是作为一个汇编的文件或文件集进入 CLR 的。 .NET 平台上语言的执行过程如图 1.4 所示。

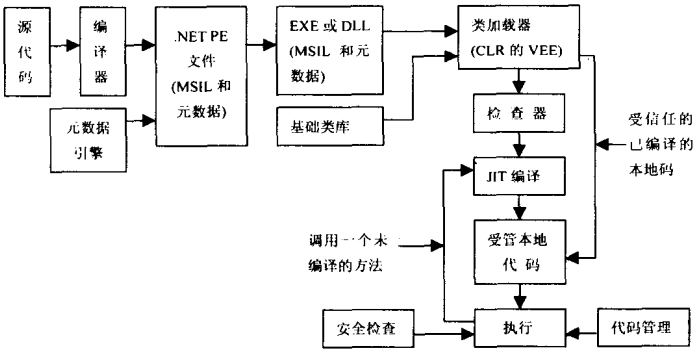


图 1.4　CLR 运行图

可以用多种语言编写源代码，然后用 .NET 编译器(所有能在 .NET 环境中编译代码的编译器都称为 .NET 编译器)编译此代码，编译器输出的不是可执行代码，而是一种特殊的伪代码，这些伪代码被称为 MSIL 代码，它就是应用程序的汇编文件。这个汇编文件包含有详细描述 MSIL 代码正确执行所需的各种相关数据类型的元数据，最后还包括一个清单，该清单中列出了所有文件和软件组件的文档，该文档指出了 CLR 在哪里可以找到具有应用程序运行所需组件的其他汇编。

为了加载一个应用程序，CLR 使用汇编的清单确定应用程序所需汇编的正确版本。然后 CLR 检查应用程序的全部汇编——MSIL 代码和描述它的元数据，从而确认代码是“类

型安全”的。这表明它只执行对恰当数据类型的恰当操作(也就是说，它不会允许开发者使用一个整数作为一个指针)，而且它只访问经过授权可以访问的内存空间。

接下来 CLR 加载应用程序的汇编中的 MSIL，并在此过程中，收集有关汇编的“证据”，例如：

- 它是从哪儿下载或安装的。
- 它需要执行什么功能(例如它是否需要写文件或发 E-mail)。
- 什么用户要运行它。
- 汇编是否拥有可信任的开发者的数字签名，以及进行数字签名后汇编是否有改动等。

这些“证据”构成了 .NET Framework 的安全要素，使得 CLR 可以判断是否运行应用程序，以及运行时需要什么许可。

当需要运行该程序时，CLR 的即时编译器(Just-In-Time，简称 JIT)再把其相应的 MSIL 翻译为可执行代码。其翻译过程是：当 .NET 运行程序时，CLR 才激活 JIT 编译器，JIT 编译器把 MSIL 翻译成本地可执行代码。因此，.NET 程序(无论是何种语言编写的)实际上是以本地代码运行的，这意味着程序的运行速度几乎与最初就把它编译成本地代码一样快，但事先编译成 MSIL 却使该程序获得了可移植性。

CLR 可以监控翻译代码的运行，并且定期清空应用程序释放的内存(使用一个称为“碎片整理”的进程)。

语言已变成一种“个人生活方式的选择”，程序员不必放弃自己喜爱的语言或不断去学习自己不擅长的语言，可以通过这种通用语言运行环境跨越平台、跨越环境。

小资料：

(1) 从本质上说，MSIL 定义了一种可移植的汇编语言。通过 .NET Framework 的工具 ILDasm 可以查看中间语言的代码，而通过 DumpBin(在 Visual Studio 下)工具可以打开可执行文件的文本文件。例如，我们编译一个文件 Hello.cs，命令如下：

```
csc  Hello.cs
```

这将生成一个可执行文件 Hello.exe，可以使用 DumpBin: dumpbin /all Hello.exe >Hello.txt，打开 Hello.txt 文件。

(2) 即时编译器 JIT 按照程序执行的需求，将 MSIL 代码翻译成本地机的执行代码。

- 编译器首先将各类用 .NET 上支持的语言(C#、Visual Basic .NET、Visual C++等)编写的源代码编译成托管的中间语言 MSIL 代码(不是机器码)，这个 MSIL 就构成可移植执行 .exe 文件(Portable Executable，简称 PE)。在编译器将源代码编译成 MSIL 的同时，元数据引擎也产生元数据信息，这些代码也可和其他语言编译的代码连接为一个 EXE 或 DLL 文件(通过链接器)。
- 由于本地的 CPU 不能直接执行 MSIL 指令。当执行应用程序时，首先类加载器将应用程序的汇编(MSIL 代码和元数据)加载到内存中，然后使用其中的元数据加载任何应用程序所需的组件支持的汇编并进行类型安全和版本检查。例如，它可能加载一个桌面应用程序所需的图形用户接口(GUI)控制的汇编。
- CLR 并不是将应用程序的所有 MSIL 代码都翻译成 CPU 指令代码，仅当用户要运行一个托管的应用程序时，操作系统装载器才加载 CLR，这时 CLR 才开始翻译该应用的 MSIL 代码。当然，若开发人员希望在应用程序首次安装到计算机中时就全部从 MSIL 转变为本机代码，则可用 PreJIT 编译器实现，PreJIT 自动把 MSIL 转换为本机代码。