从基础到实践丛书

C#编程从基础到实践

殷泰晖　张强　杨豹　等编著

配套光盘提供了本书所有实例程序的源代码

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

从基础到实践丛书

C#编程

从基础到实践

殷泰晖　张强　杨豹　等编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书内容分为 7 部分，共 18 章。第 1 部分是 C#编程基础，简要介绍 C#与.NET 的关系、C#开发及运行环境，以及 C#的语言基础，包括数据类型与常变量、运算符和表达式、程序流控制等内容。第 2 部分介绍面向对象的 C#程序设计，内容包括面向对象的思想、类和对象相关介绍，以及继承机制与方法调用等内容。第 3 部分引入 C#编程的接口、文件和流操作的相关内容，包括接口定义与实现、抽象类与接口、文件读写和流的操作。第 4 部分开始介绍 C#网络编程基础，内容包括网络通信量的监视、TCP 和 UDP 编程简介、DNS 的使用和套接字编程简介等。第 5 部分深入介绍了 C#套接字编程的相关内容，包括面向连接的 TCP 协议编程、无连接的 UDP 协议编程、异步套接字编程和网络组播技术等内容。第 6 部分介绍大量的分类开发实例的实现，包括 DNS 的开发、FTP 协议编程、SMTP 协议编程、HTTP 协议编程和 ASP 网页应用程序编程的实现。第 7 部分介绍一个大型的 C#网络开发实例的实现。

本书可作为初学者学习 C#网络程序设计的入门教程，也可以作为计算机网络设计人员与开发人员的技术参考书。

图书在版编目（CIP）数据

C#编程从基础到实践 / 殷泰晖等编著. —北京：电子工业出版社，2007.6
（从基础到实践丛书）
ISBN 978-7-121-04212-6

Ⅰ. C… Ⅱ. 殷… Ⅲ. C 语言－程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2007）第 049741 号

责任编辑：韩 明
印　　刷：北京智力达印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：30　字数：750 千字
印　　次：2007 年 6 月第 1 次印刷
印　　数：5000 册　定价：49.00 元（含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

序 言

C#语言是由微软公司的 Anders Hejlsberg 和 Scott Wiltamuth 领导的开发小组专门为.NET 平台设计的语言。它从 C、C++和 Java 语言发展而来，集成了这 3 种语言的优点，并使用事件驱动和完全面向对象的编程模式。

C#语言是一种基于 XML 语言，但又增强了 Web 服务功能的运行于.NET 平台的新型语言。.NET 平台提供的工具和服务能充分发挥系统的计算和通信能力，支持.NET 的大多数框架都是用 C#编写的。并且.NET 框架平台简化了网络应用程序的设计，在编程时，.NET 库提供了许多网络程序设计需要的网络类。因为 C#具有良好的面向对象特性，使用简单的 C#语言结构，所有的组件都可以转换成 Web 服务，可以完成底层平台的调用与底层代码的控制，实现跨语言、跨平台的因特网远程调用。所以，在复杂的网络应用程序设计中，C#语言是一种较好的选择。

与 C 语言等程序设计语言相比，通过.NET 平台提供的支持，C#语言简化了网络应用程序设计的复杂性，程序设计人员只须使用或修改.NET 库提供的网络库即可实现大部分的网络功能。

本书内容

本书的内容包括 C#语言编程的基础、面向对象的相关知识、大量网络编程的相关协议，以及与之相关的.NET 平台提供的 C#相关类的介绍。因此，本书可以作为 C#网络程序设计人员的入门教程，同时也是网络程序设计人员的一本有价值的参考书。

本书内容不仅涵盖了.NET 和 C#的关系、C#的开发及运行环境、C#语言的语法基础、面向对象的设计思想，以及 C#作为一种面向对象程序设计语言的深入介绍、C#接口实现、文件和流的操作等在 C#网络应用程序开发时的必备基础知识，还对网络编程中使用的常用协议和.NET 平台支持的相关类进行了详细介绍，并提供了详细的实现实例，包括 DNS、TCP、UDP、HTTP、FTP、SMTP 等协议。

本书特色

本书的主要特点是在对 C#网络编程的相关内容进行介绍的同时，引入了大量的程序实例，并且对实现过程的重点和难点进行详细讲述，让读者通过具体的网络应用程序的实

现来充分了解 C#网络程序设计的基本要点。通过这种理论和实践相结合的方式，能够让读者在较短的时间内对 C#网络编程有较全面的掌握。

写作目的

本书的宗旨在于让读者能够在较短的时间内精通 C#网络应用程序。对于初学者，可以从内容翔实的程序实例入手，轻松掌握 C#网络程序的基本实现技术。而对于接触过 C#网络程序设计的读者，也可以从本书介绍的实例实现、注意要点等内容之中获得一些经验。

本书第 1、2、10、17、18 章由张强编写，第 3、4、7、9、12、14、16 章由殷泰晖编写，第 5、6、8、11、15 章由杨豹编写，第 13 章由龚春叶编写，全书由殷泰晖统稿。由于编者水平有限，书中难免存在错误和不妥之处，敬请广大读者谅解，并提供宝贵意见。联系邮件：elegend2007@163.com。

编　者

2007 年 2 月

目 录

第 1 部分 C#编程基础

第 1 章 C#与.NET 概述 2

1.1 NET 与 C# 2
1.1.1 Microsoft .NET 2
1.1.2 C#与.NET 5
1.2 C#开发及运行环境 6
1.2.1 C#开发工具及相关 6
1.2.2 C#安装与执行环境 9
1.3 C#编程与实例 12
1.3.1 C#编程特点与风格 12
1.3.2 使用 C# 创建.NET 应用程序 14
1.4 分析与小结 15

第 2 章 C#语言基础 16

2.1 数据类型与常变量 16
2.1.1 值类型 16
2.1.2 引用类型 19
2.1.3 常量与变量 21
2.1.4 类型转换 23
2.2 操作符和表达式 29
2.2.1 操作运算符 29
2.2.2 算术操作符和算术表达式 30
2.2.3 赋值操作符和赋值表达式 33
2.2.4 关系操作符和关系表达式 33
2.2.5 逻辑操作符和逻辑表达式 35
2.2.6 位运算 36
2.2.7 其他特殊操作符 39
2.3 程序流控 42
2.3.1 条件语句 42
2.3.2 循环语句 45
2.3.3 跳转结构 49
2.3.4 条件编译 52
2.3.5 异常处理 54
2.4 小结 55

第 2 部分 面向对象的 C# 程序设计

第 3 章 C#面向对象设计基础 58

3.1 面向对象的思想 58
3.1.1 面向对象的设计理念 58
3.1.2 面向对象的程序设计过程 59
3.2 类和对象 59
3.2.1 类和对象的关系 60
3.2.2 类 60
3.2.3 构造函数和析构函数 65
3.3 其他相关介绍 68
3.3.1 方法 68
3.3.2 域的概念 74
3.3.3 属性 78
3.3.4 索引指示器 82
3.3.5 事件 84

3.4 小结 ····· 89

第 4 章 C#面向对象深入分析 ····· 91

4.1 C#中的继承与多态 ····· 91
4.1.1 继承机制 ····· 91
4.1.2 访问与隐藏基类成员 ····· 94
4.1.3 关于继承中的一些问题 ····· 101
4.1.4 多态性与虚方法 ····· 106
4.2 C#类型转换 ····· 110
4.2.1 隐式类型转换 ····· 110
4.2.2 显式类型转换 ····· 111
4.2.3 类的引用转换 ····· 112
4.2.4 装箱和拆箱 ····· 113
4.3 Object 基类 ····· 114
4.3.1 Object 类中的方法 ····· 114
4.3.2 使用 Object 类中的方法 ····· 115
4.3.3 重载 Object 类中的方法 ····· 116
4.4 小结 ····· 117

第 3 部分 C#编程提高

第 5 章 接口 ····· 120

5.1 接口的定义与实现 ····· 120
5.1.1 接口的定义 ····· 120
5.1.2 接口的成员 ····· 122
5.1.3 接口的实现 ····· 128
5.2 抽象类与接口 ····· 147
5.2.1 抽象类 ····· 147
5.2.2 抽象类和接口 ····· 148
5.3 小结 ····· 149

第 6 章 文件和流 ····· 150

6.1 文件、目录和流 ····· 150
6.2 文件操作 ····· 151
6.2.1 文件的创建 ····· 151
6.2.2 文件读写 ····· 156
6.2.3 异步文件操作 ····· 164
6.3 目录和路径操作 ····· 170
6.3.1 Directory 类 ····· 170
6.3.2 DirectoryInfo 类 ····· 172
6.3.3 Path 类 ····· 174
6.4 小结 ····· 176

第 4 部分 网络编程基础

第 7 章 网络编程相关内容 ····· 178

7.1 网络通信监视 ····· 178
7.1.1 网络通信监视介绍 ····· 178
7.1.2 网络数据包分析 ····· 183
7.2 TCP 编程和 UDP 编程 ····· 185
7.2.1 TCP 编程特点 ····· 185
7.2.2 UDP 编程特点 ····· 186
7.2.3 获取网络配置信息 ····· 187
7.3 C# DNS 编程实例 ····· 192
7.3.1 域名系统 DNS ····· 193
7.3.2 Windows DNS 客户端信息 ····· 196
7.3.3 在 C#网络编程中获取 DNS 信息 ····· 200
7.4 小结 ····· 208

第 8 章 C#网络套接字编程 ····· 209

8.1 套接字编程技术简介 ····· 209
8.1.1 UNIX 中的套接字编程 ····· 209
8.1.2 Windows 中的套接字编程 ····· 214
8.2 C#套接字编程 ····· 216
8.2.1 处理 IP 地址 ····· 217
8.2.2 C#套接口 ····· 220
8.3 C#套接字助手类 ····· 224
8.3.1 TcpClient ····· 224
8.3.2 TcpListener ····· 226
8.3.3 UdpClient ····· 229

8.4 小结 ……233

第 5 部分 网络编程深入分析

第 9 章 面向连接的套接字编程 ……236

9.1 C# TCP 协议编程基础 ……236
9.1.1 TCP 协议相关类简介 ……236
9.1.2 若干注意要点 ……242
9.2 简单的 TCP 服务器 ……243
9.2.1 TCP 服务器实例 ……243
9.2.2 实例分析 ……245
9.2.3 服务器监视 ……246
9.3 简单 TCP 客户端程序 ……246
9.3.1 TCP 客户端程序实例 ……246
9.3.2 相关测试 ……247
9.4 在 TCP 通信中使用 C#流 ……248
9.4.1 NetworkStream 类 ……248
9.4.2 StreamReader 和 StreamWrite 类 ….252
9.4.3 C#通信流的实现 ……254
9.5 小结 ……256

第 10 章 无连接套接字编程 ……257

10.1 UDP 应用简述 ……257
10.1.1 UDP 的几个特点 ……258
10.1.2 UDP 与 TCP 区别 ……259
10.1.3 UDP 的应用范围 ……259
10.1.4 UDP 程序段分析 ……260
10.1.5 UDP 信息区分 ……264
10.2 C#中 UDP 通信过程 ……266
10.2.1 处理 UDP 数据丢失 ……266
10.2.2 处理报文丢失 ……269
10.2.3 数据报重传 ……273
10.3 C#中 UDP 实现机理 ……278
10.3.1 Socket 详述 ……278
10.3.2 C#中 UDP 通信的函数 ……279
10.3.3 UDP 套接字小结 ……282

第 11 章 异步套接字编程 ……283

11.1 Windows 事件驱动编程 ……283
11.1.1 事件和委托 ……283
11.1.2 AsynCallback 类 ……286
11.2 异步套接字 ……286
11.2.1 面向连接的异步套接字方法 ……287
11.2.2 无连接的异步套接字方法 ……298
11.3 非阻塞套接字方法 ……299
11.3.1 轮询方法 Poll() ……300
11.3.2 选择方法 Select() ……303
11.4 小结 ……305

第 12 章 C#网络组播技术 ……306

12.1 组播定义及说明 ……306
12.1.1 什么是组播 ……306
12.1.2 使用组播特点发送报文 ……310
12.2 C# IP 组播 ……313
12.2.1 C#IP 组播支持 ……313
12.2.2 关键实现技术 ……316
12.3 组播应用程序例子 ……323
12.3.1 一个组播应用实例 ……324
12.3.2 实例分析及总结 ……327
12.4 小结 ……328

第 6 部分 C#分类开发分析

第 13 章 C#DNS 开发 ……332

13.1 域名系统 DNS ……332
13.1.1 DNS 结构 ……332
13.1.2 DNS 主机发现 ……333
13.2 DNS 相关类介绍 ……334
13.2.1 DNS 同步和异步方法 ……334
13.2.2 DnsPermission 类 ……336

13.3 C# DNS 服务系统……342
13.3.1 DNS 服务实现源码……342
13.3.2 程序分析……354
13.4 小结……355

第 14 章 C# FTP 编程……356

14.1 FTP 相关……356
14.1.1 FTP 简介……356
14.1.2 FTP 文件传输原理……358
14.2 FTP 开发过程……360
14.2.1 FTP 服务流程……360
14.2.2 典型的 FTP 过程……361
14.3 FTP 文件传输开发实例……362
14.3.1 简单的 FTP 实现案例……362
14.3.2 案例分析……377
14.4 小结……384

第 15 章 SMTP 协议编程……385

15.1 SMTP 协议简介……385
15.1.1 E-mail 基础……385
15.1.2 SMTP 和 Windows……388
15.2 SMTP 协议编程……389
15.2.1 SmtpMail 类……389
15.2.2 扩展 Mail 信息的使用……390
15.3 POP3 客户端……395
15.3.1 POP3 协议……395
15.3.2 POP3 客户端……397

第 16 章 C# HTTP 协议编程……403

16.1 C# HTTP 编程……403
16.1.1 HTTP 协议简介……403
16.1.2 与 HTTP 相关类的介绍……406
16.2 HTTP 协议编程常用方法详解……416
16.2.1 HTTP 的方法调用与实现……416
16.2.2 使用 HTTP 实现 Web 服务……418
16.2.3 Internet 的请求和响应……421
16.3 HTTP 协议编程实例……424
16.3.1 一个典型 HTTP 例子……424
16.3.2 C#实例应用及分析……426
16.4 小结……426

第 17 章 ASP 页面应用程序开发……428

17.1 ASP.NET 基础……428
17.1.1 ASP.NET 的几个特点……428
17.1.2 ASP.NET 编程体验……429
17.1.3 ASP.NET 体系结构……432
17.2 ASP.NET 的产生与演化……437
17.2.1 传统 ASP 的缺陷……437
17.2.2 ASP.NET 的设计目标……438
17.2.3 ASP.NET 的设计方法……438
17.3 ASP.NET 页面编写……445
17.3.1 事件与过程……445
17.3.2 示例分析……446

第 7 部分 大型 C#综合网络开发实例

第 18 章 C#开发综合实例——网络学籍管理系统……450

18.1 系统开发环境……450
18.1.1 开发与部署环境……450
18.2 需求分析及设计说明……450
18.2.1 需求分析……450
18.2.2 总体设计……452
18.2.3 模块功能分析……454
18.2.4 数据库定义……455
18.3 网络学籍系统……455
18.3.1 网络学籍系统实现框架……456
18.3.2 网络学籍系统主要源码……458
18.4 分析与总结……468

PART 1

第 1 部分 C#编程基础

- 第 1 章　C#与.NET 概述
- 第 2 章　C#语言基础

第 1 章　C#与.NET 概述

欢迎您学习 C#（读做 C-Sharp）!

C#语言是由微软公司的 Anders Hejlsberg 和 Scott Wiltamuth 领导的开发小组专门为.NET 平台设计的语言。它从 C、C++和 Java 语言发展而来，集成了这 3 种语言最优秀的特点，并使用事件驱动，完全面向对象的编程模式。本书在介绍 C#和.NET 的基本知识以后，将主要讲述 C#网络编程及相关内容。

本章内容包括：

- Microsoft .NET 介绍；
- C#开发及运行环境说明；
- C#编程特点与风格阐述；
- 一个小 C#程序的编写及编译运行过程。

1.1　NET 与 C#

.NET（读做 dot-net）作为一种如今广泛应用的软件开发平台，允许用完全不同的编程语言编写的应用程序相互通信；这个平台也允许开发者开发基于 Web 的应用程序，这些应用程序可以分布在桌面计算机等大量的设备（甚至包括无线设备）上。

1.1.1　Microsoft .NET

1．什么是.NET

何谓.NET？.NET 是 Microsoft 公司面向 XML Web（下一代软件）服务的平台，该平台将信息、设备和人以一种统一的、个性化的方式联系起来。借助.NET 平台，可以创建和使用基于 XML 的应用程序、进程和 Web 站点，以及服务，它们之间可以按设计在任何平台或智能设备上共享和组合信息与功能，以向单位和个人提供定制好的解决方案。.NET 包含了建立和运行基于 XML 的软件所需要的全部部件。

.NET 是一个全面的产品家族，它建立在行业标准和 Internet 标准之上，提供开发（工具）、管理（服务器）、使用（构造块服务和智能客户端）和 XML Web 服务体验（丰富的用户体验）；.NET 使用分布式计算模型并基于开放标准（如 XML）将 PC 与其他智能设备连接在一起。它作为下一代的 Internet 计算模型，各个 XML Web 服务之间彼此是松耦合的，通过 XML 进行必要通信，协同完成某一特定的任务。

Microsoft .NET 将从根本上改变我们的思考方式和使用电脑的方式。目前，“服务器”和“桌面电脑”这两种概念占据了计算领域的统治地位。然而，.NET 是一种分布式计算范例，它没有传统上的服务器和桌面电脑的区别，取而代之的是计算的处理被放在最合适的地方进行，可能是服务器或 PC，也有可能是手提电脑和其他智能设备，这就是智能计算。

.NET 定义了一种公用语言子集（Common Language Subset，CLS），这是一种为其规范的语言与类库之间提供无缝集成的混合语。Microsoft .NET 统一了编程类库，提供了对

下一代网络通信标准和可扩展标记语言（Extensible Markup Language，XML）的完全支持，从而使得程序开发变得简单容易。.NET 作为一种面向网络，支持各用户终端的开发平台环境，其核心目标之一就是要搭建第三代因特网平台，解决网络之间的协同合作问题，最大限度地获取信息，提供尽可能全面的服务。

2．使用.NET 的优点

Microsoft .NET 为我们带来的好处可以从下列方面得到体现。

（1）相对来说，程序员是比较缺乏的，因此雇用的费用很高。然而，Microsoft .NET 使编程工作变得更加容易，开发投资的回报率也趋最大化。开发者们可以创建能重用的 XML Web 服务，而不再是一个单一的程序。这些 Web 服务易于编程和调试，彼此之间相互独立，通过 XML message 进行通信及合作。

（2）Microsoft .NET 减少了程序员要写的代码量。一个 XML Web 服务能适用于所有的设备，不必再去为每一个设备编写一个不同的版本。另外，将显示特性与.NET 体验分开，以便以后加入新的接口技术，比如，可以增加语音或手写识别功能，而不必去重写程序。

（3）Microsoft .NET 开创了全新的商业模型，它使得一个公司可以用多种方法来把自己的技术商品化。技术提供商可以把他们现有的软件包转变为 XML Web 服务，并把这些服务出售给需要这些功能的第三方，或供给.NET 体验提供商，用以构建新的软件包。

（4）Microsoft .NET 允许 IT 部门使用其他提供商的 XML Web 服务，减少内部研发的开销，并能提高工作效率。

（5）Microsoft .NET 对“用户界面友好”作了重新定义。终端用户能够徜徉于一个智能化的、个性化的 Internet 上，它能记住用户的个人设置，并在适当的时候，向用户使用的智能设备发送适当的数据。

3．.NET 的核心组件

NET 的核心组件包括：

（1）一组用于创建因特网操作系统的构建块，其中包括 Passport.NET（用于用户认证）和用于文件存储的服务、用户首选项管理、日历管理，以及众多的其他任务。

（2）构建和管理新一代服务的基本结构和工具，包括 Visual Studio.NET、.NET 企业服务器、.NET 框架和 Windows.NET。

（3）能够启用新型智能因特网设备的.NET 设备软件。

（4）NET 用户体验。

具体来说，C#这种微软提出的新语言在协同了.NET 基本类库后，具有了下面所列的特征。

（1）支持名字空间

对名字空间的支持是出于网络集成的需要。名字空间实际上提供了一种对象分类和寻址的方式。

（2）分布式计算的代理技术

分布式计算涉及构件的交互，这必然要求语言提供事件处理的机制，其中包括显示的事件通知；其次，对象在网络上传输还要涉及异构平台问题，这要求语言支持对象封装。

（3）支持垃圾回收

在如今的 Internet 环境中，某些隐蔽性的潜在错误可能导致程序不可想象的问题，因此必须提供一个安全的编程方式。.NET 的公共语言运行时提供了垃圾回收等各类安全机制，从而使开发者更加容易地编写和维护解决复杂业务问题的程序。

（4）统一类型系统的支持

为了保证语言的互操作性，C#语言同统一类型系统保持了一致性。

（5）真正的数据隐藏

网络计算应该通过方法，实现真正意义上的数据隐藏，这就要求语言支持性质和接口的特性，C#语言保证了数据的隐藏性。

（6）支持构件技术

构件对象模型（COM）一直是最为成功的可重用软件模型，但在过去，开发构件很不方便，因为 COM 要求开发人员必须手动处理一些复杂问题，比如，清空不用构件曾占用的内存，计算构件的使用次数，建立或撤销线程和进程，以及处理版本控制问题等。COM 开发过程中最难的一件事就是处理 COM 基本结构。.NET 框架的一个主要目的就是使得 COM 开发更加容易。因此，.NET 框架自动处理了所有在开发人员看来是与 COM 紧密相关的任务，包括引用计算、接口描述和注册，从而简化了构件的开发。

（7）支持 XML 技术

XML 是.NET 平台上最重要的特征，也是实现 Web 服务的关键。

（8）支持持久数据

网络计算的数据需要存放起来，所以语言必须支持数据的持久性存储。C#语言具备这一特性。

（9）支持本地代码

在应用中，在考虑性能问题或与现存的应用程序接口的问题时，要用到“本地”代码。过去为了解决这样的问题，开发者只能使用 C++来开发，但由于 C++本身的开发效率不高，所以新的语言应该在支持快速开发的同时又要支持对本地代码的调用，这样才有利于本地编程者的开发。而很多本地码的操作都是底层的，不可避免要涉及系统服务。为了保证代码运行的安全，语言要提供安全机制，而且要能够处理受管代码和非受管代码的交互。C#在这方面做得较好。

（10）跨语言集成

语言要支持跨语言互用特性，首先就必须遵守 CTS 和 CLS 的约定，其次，该语言应该可被编译成为中间代码，以同其他语言进行交互操作。C#具备了这样的特点。

（11）版本问题

软件的版本问题一直是让人们头疼的问题。尤其在 Internet 上进行分布式开发的今天，软件版本问题的重要性更加突出。C#提供了内置的版本支持。

4. 分析.NET 的继承性

尽管 Microsoft .NET 给计算带来了翻天覆地的变化，但还有很多东西依然没有得到改变。

（1）终端用户将依然使用熟悉的界面，就像.NET 体验中的 Microsoft Office 一样。这可以减少再培训的开支，也意味着用户可以马上开始使用.NET 软件。

（2）硬件上运行的还是像 Windows、UNIX、Windows CE 和 Palm OS 一样的操作系统。实际上，.NET 增加了软件的运行场所，但同时减少了开发的负担。由于 XML Web 服务只使用 XML 与设备通信，所以任何智能设备都可以享用 XML Web 服务。

（3）对程序员来说，他们依然可以使用他们原先熟悉的编程语言。.NET 平台借助于.NET 框架的公共语言运行时间库（CLR）使得用不同语言开发的 XML Web 服务之间也可以相互操作。有没有.NET 体验问题不大，你依旧可以用 Visual Basic、Java，甚至 COBOL 创建 XML Web 服务。

（4）原先系统无须被替换。一部分的 Microsoft .NET 产品就是为了能方便地将现有的系统整合到新的 XML Web 服务和.NET 体验中去而设计的。比如 BizTalk Server，它管理的商务流程（Business Process Orchestration）包括了对现有系统和数据格式的支持，并会执行一些必要的转换，将数据转成 XML。

所以，这种下一代的分布式计算是建立在目前这一代基础上的。Microsoft .NET 不是我们所想象的那样，对现在的应用软件做了大规模的替换，而是一个自然的进化过程，它在原先的技术孤岛之间建立了协作关系，协同工作能力逐渐加强，我们也将从中受益无穷。

1.1.2　C#与.NET

Microsoft 公司是这样描述 C#的：“C#是从 C 和 C++派生来的一种简单、现代、面向对象和类型安全的编程语言。C#主要是从 C/C++编程语言家族移植过来的，C 和 C++的程序员会马上熟悉它。C#试图结合 Visual Basic 快速开发的能力和 C++强大灵活的能力。”

长期以来，C 和 C++一直是很有生命力的程序设计语言，它们为程序员提供了丰富的功能、高度的灵活机制和强大的底层控制能力。但是，这一切都不得不在效率上做出一定的牺牲。由于 C 和 C++为我们带来的高度灵活性，使得我们不得不忍受对其学习的艰苦和开发的难度。正因为如此，程序员们试图寻求一种在开发能力和使用效率上有更好平衡的语言。而且，更重要的是，它要与当前的 Web 应用有很好的结合。

C#就是这样一种由 Microsoft 开发的新型编程语言，它具有现代的面向对象的特点，这使得程序员能在新的微软.NET 平台上快速开发种类丰富的应用程序。由于它是从 C 和 C++中派生出来的，因此具有 C++的功能。同时，由于是 Microsoft 公司的产品，它又同 VB 一样简单。对于 Web 开发而言，C#像 Java，同时具有 Delphi 的一些优点。Microsoft 宣称：C#是开发.NET 框架应用程序的最好语言。

C#是.NET 的关键性语言，它是整个.NET 平台的基础。与 C#相比，.NET 所支持的其他语言显然是配角身份。比如，VB.NET 的存在主要是对千万个 VB 开发人员负责。对于 JScript.NET 和 Managed C++也同样可以这么说，后者只是增加了调用.NET 类的 C++ 语言。C#是唯一没有在设计思路中加入前辈语言某种遗传的新事物。

.NET 平台将 C#作为其固有语言，重温了许多 Java 的技术规则。C#中也有一个虚拟机，叫做公用语言运行环境（CLR），它的对象也具有同样的层次。但是 C#的设计意图是使用全部的 Win32 API，甚至更多。由于 C#与 Windows 的体系结构相似，因此 C#很容易被开发人员所熟悉。

C#本质上是 C++的进化产物，使用了包括声明、表达式及操作符在内的许多 C++特征，但是，C#还有更多的增强功能，比如类型安全（type-Safe）、事件处理、碎片账集、代码

安全性等。在C#中，除了可以使用许多API外，更能使用.NET类。特别地，我们可以处理COM的自动化和C类型的函数。C#还让你调用无管理的代码，也就是在CLR引擎控制之外的代码。这种不安全的模式允许你操作原始指针来读和写内置碎片账集控制以外的内存。

1.2 C#开发及运行环境

1.2.1 C#开发工具及相关

C#是运行于.NET平台之上的，其各种特性与.NET紧密相关。它本身没有运行库，其许多强大的功能有赖于.NET平台的支持。因此，这里介绍C#的运行环境，也必须涉及.NET的公用语言运行时环境、语言规范和它的开发工具。

.NET主要包含下面4个组成部分。

（1）虚拟对象系统（VOS）；

（2）元数据；

（3）公用语言规范；

（4）虚拟执行系统。

1. 虚拟对象系统

.NET 跨语言集成的特性来自于虚拟对象系统的支持。在不同语言间进行代码复用和应用集成的最大问题在于不同语言类型系统间的相容性。因为虽然不同语言语法结构大致相同，但数据类型与语言环境本身的各种特点紧密联系，因此，相同的数据类型在不同的语言环境下表示的意义也存在差别。一个简单的例子，比如对于整数类型，在VB中是16位的，可在MSSQL中长度却是32位的。这就造成了互相之间的不兼容。

VOS的作用就是为了改变此种状况。它既支持过程性语言，也支持面向对象语言，同时它提供了一个类型丰富的系统来容纳其所支持的各种语言特性，从而在最大限度上屏蔽了不同语言类型间的转换，使程序员可以自由地选择自己喜欢的语言进行开发，保证了不同语言间的兼容与集成。

对于面向对象的语言，它统一了不同编程语言的对象类型，每一个对象在VOS中都被唯一标识，以区别于其他对象；而对于过程型语言，它描述了值的类型，并指定了类型的所有值必须遵守的规则。

2. 元数据

元数据是对VOS中类型描述代码的一种称呼。在编译程序将源代码转换为中间代码时，它自动生成并与编译后的源代码共同包含在二进制代码文件中。元数据携带了源代码中类型信息的描述，程序使用的类型描述与其自身绑定在一起，这在一定程度上解决了版本问题。

在定位与装载类型时，系统通过读取并解析元数据来获取应用程序中的类型信息，编译器获得加载的类型信息后，将中间语言代码翻译成本地代码，在此基础上根据程序或用户要求建立类型实例。元数据在解决方法的调用和建立运行期上下文界限等方面都有着自己的作用，而关于元数据的一切都是由.NET在后台完成的。

3. 公用语言规范

.NET 运行时集成了各种语言，允许一种语言使用其他语言的对象，而使得这成为可能的是运行时的类型系统、元数据和公共执行环境。然而，语言集成的任务很艰巨，因为语言之间的差异太大，并且大部分编程语言一般都不支持所有的运行时功能。比如，有些语言不区分大小写敏感，不提供无符号整数，不提供运算符重载，不提供联合类型，不提供可变参数方法等。但是，作为语言集成的必要条件就是，如果某语言创建的类型想被其他编程语言访问，那么它必须使用其他语言中同样有的特性。

有鉴于此，微软定义了公用语言规范（Common Language Specification，CLS）。我们可以将公共语言运行时作为许多编程语言功能的并集，而将 CLS 作为这些功能的子集。该子集是许多编程语言共有的功能集合，它作为 CLR（公用语言运行环境）定义的语言特性集合，主要用来解决互操作问题。

CLR 使得设计跨语言的组件与应用变得更加容易，因为以不同语言设计的对象间能够彼此进行通信，并且它们的行为能够紧密地综合与协调。比方说，我们定义了一个类，然后我们就可以在另一种不同的语言中从该类中派生一个类，或者调用其中的一个方法。我们也可以向另一种语言中类的方法传递该类的一个实例。这种跨语言的集成之所以可能，是因为以运行时间为目标的语言编译器与工具使用一种运行时间所定义的公用类型系统，它们遵守运行时的规则（公用语言规范）来定义新的类型，生成、使用、保持并绑定类型。

因为 CLS 专门处理语言互操作性的问题，其规则只适合“外部可见的”类型。CLS 认为语言的互操作性仅仅在组装之间重要，也就是说，在一个组装内对语言没有什么要求，因此，CLS 的规则只适合在组装内的“外部可见的”类型。下面简单地列出 CLS 的 40 条规则。

（1）这些规则只适合类型暴露在组装内的“外部可见的”部分。

（2）装匣类型不在 CLS 的范围内，但是利用 System.Object、System.ValueType 或 System.Enum 这些类型是可以的。

（3）语言必须遵守 Unicode 3.0 标准。

（4）语言必须提供一种机制来把那些在语言中的关键字当作标识符使用。而且语言应该提供一种机制来把那些在语言中的关键字当作方法名使用。

（5）CLS 遵从的语言中，所有的名字都不能重复，除非在重载情况下可以相同。

（6）字段和嵌套类型只能通过标识符来比较，而方法、事件和性质则除了比较返回值，还要比较参数等其他方面。

（7）枚举的底层类型必须是内建的整型。

（8）有两类枚举，通过是否有 System.FlagsAttribute 定制属性来区别。一个代表着命名的整型，另一个代表着命名的置标位。

（9）枚举的静态字段本身必须有枚举类型。

（10）在重载一个可继承的方法时，其可访问类型不能被修改。但是，如果是从另一个组装中的方法继承时，则可以修改。此时，重载必须具有访问修饰符。

（11）不允许创建嵌套类型。

（12）不允许类型引用。

（13）型构中的所有类型都必须是 CLS 遵从的。

（14）如果成员可见并可访问，那么成员的型构也可见并可访问。比如，一个在外部可见的方法不可包含一个外部不可见的参数。

（15）CLS 没有变参限制，其所支持的调用转换是标准的调用转换。

（16）数组元素的类型必须是遵从 CLS 的，有固定的维数，每一维以零为起点。System.Array 是 CLS 遵从的抽象类型，但从它继承的类型也有些不是 CLS 遵从的。

（17）非受管指针类型不是 CLS 遵从的。

（18）CLS 遵从的工具必须意识到，一个类可能实现两个接口，而这两个接口可能包含同名同型构的方法。这两个方法被认为是不同的，不需要同等实现。

（19）为了实现 CLS 遵从的接口，不允许包含非 CLS 遵从的方法。

（20）CLS 遵从的方法不定义静态或实例方法，也不定义数据字段。它只定义性质、事件和虚方法。

（21）CLS 遵从的类、值类型和接口不许包含非 CLS 遵从的接口。

（22）在实例数据被访问之前，对象的构造子必须调用某些基类的构造子。注意，这不适用于值类型。

（23）对于对象引用类型，除了创建对象，其他时候不得调用构造函数，而且一个对象不得初始化两次以上。

（24）CLS 遵从的类不许继承自同样 CLS 遵从的类。

（25）实现了性质的 Get 和 Set 方法的方法，必须在元数据中以 mdSpecialName 标注。

（26）性质的访问能力和它的访问子必须保持一致。

（27）性质及其访问子必须全是静态的、虚的或实例化的。

（28）性质的类型必须和 Getter 返回的类型相同，也必须与 Setter 方法的最后一个参数类型相同。性质的参数类型就是 Getter 和 Setter 方法的参数类型。所有的类型都必须是 CLS 遵从的，也不能是受管指针。

（29）性质必须遵守特定的命名规范。

（30）事件的 Add 和 Remove 方法必须成对出现。

（31）事件的 Add 和 Remove 方法必须带一个与事件类型相同的类型参数。该类型派生自 System.Delegate。

（32）事件的命名必须遵守特定的命名规范。

（33）CLS 遵从的工具必须处理一套自定义属性的编码。

（34）CLS 遵从的工具不应生成外部可见所需的修饰符，应略去不懂的修饰符。

（35）全局静态变量和方法是 CLS 不允许的。

（36）类型与包含它的组装的 CLS 遵从性不同的时候，类型必须以 System.CLSCompliant-Attribute 标注。成员和类亦是如此。

（37）只有性质和方法可以被重载。

（38）性质、实例和虚方法值可以根据它们的参数个数和类型来重载，除了使用转换操作符 op_Implicit 和 op_Explicit，它们本身可以根据返回值被重载。

（39）如果 op_Implicit 和 op_Explicit 被重载，那么必须提供限制语义。

（40）抛出的异常对象必须是 System.Exception，或从其继承而来。

4. 虚拟执行系统

虚拟执行系统（Virtual Execution System，VES）是 VOS 的实现，它用来驱动运行环