Wrox Programmer to Programmer™

BUILD WINDOWS 8 STYLE APPS WITH C#

.NET开发经典名著

Professional C# 2012 and .NET 4.5

# C#高级编程（第8版）


Christian Nagel
Bill Evjen
[美] Jay Glynn 著
Karli Watson
Morgan Skinner
李铭 译
黄静 审校


清华大学出版社

.NET 开发经典名著

# C#高级编程(第8版)


Christian Nagel
Bill Evjen
[美] [redacted] 著
Morgan Skinner
李 铭 译



清华大学出版社

北 京

Christian Nagel，Bill Evjen，Jay Glynn，Karli Watson，Morgan Skinner
Professional C# 2012 and .NET 4.5
EISBN：978-1-118-31442-5


图书在版编目(CIP)数据
C#高级编程(第 8 版)/(美)内格尔(Nagel, C.) 等著；李铭 译. —北京：清华大学出版社，2013.10
(.NET开发经典名著)
书名原文：Professional C# 2012 and .NET 4.5
ISBN 978-7-302-33411-8
Ⅰ. ①C…　Ⅱ. ①内…　②李…　Ⅲ. ①C 语言－程序设计　Ⅳ. ①TP312
中国版本图书馆 CIP 数据核字(2013)第 180825 号

责任编辑：王　军　于　平
装帧设计：牛艳敏
责任校对：成凤进
责任印制：杨　艳

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
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：96.75　插　页：1　字　数：2663 千字
版　　次：2013 年 11 月第 1 版　　印　次：2013 年 11 月第1 次印刷
印　　数：1～5000
定　　价：148.00 元

产品编号：050389-01

# 作者简介

**Christian Nagel** 是 Microsoft 区域董事、Microsoft MVP，thinktecture 的合作伙伴，CN 革新技术的拥有者，他是一位软件架构师和开发人员，为开发 Microsoft .NET 解决方案提供培训和咨询服务。他具备超过 25 年的软件开发经验。Christian 从 PDP 11 和 VAX/VMS 系统开始其计算机生涯，熟悉各种语言和平台。自从 2000 年以来，(那时.NET 还只是一个技术框架)，他就开始使用各种.NET 技术构建大量.NET 解决方案。目前，他主要开发 Windows Store 应用程序来访问 Windows Azure 服务。他具备 Microsoft 技术的深厚功底，编写了大量图书，并获得了 Microsoft 认证培训师和专业开发人员证书。Christian 在国际会议上发表演讲(如 TechEd 和 Tech Days)并创立 INETA Europe，以支持.NET 用户组。通过 Web 站点 www.cninnovation.com 和 www.thinktecture.com 可以联系 Christian，其微博是 @christiannagel。

**Jay Glynn** 开发软件的时间超过 20 年，使用 PICK Basic 为 PICK 操作系统编写应用程序。到目前为止，他使用过 Paradox PAL and Object PAL、Delphi、VBA、Visual Basic、C、Java 和 C#编写软件。他目前是 UL PureSafety 的高级软件工程师，编写基于 Web 的应用程序。

**Morgan Skinner** 年轻时对 Sinclair ZX80 很感兴趣，在校期间就开始了计算机生涯，当时他对教师编写的一些代码不感兴趣，便开始用汇编语言编程。从此以后他使用各种语言和平台，包括 VAX 宏汇编程序、Pascal、Modula2、Smalltalk、X86 汇编语言、PowerBuilder、C/C++、VB 和目前的 C#，自从 2000 年发布 PDC 以来，他就用.NET 编程，而且非常喜欢.NET，于是在 2001 年加入了 Microsoft。他现在是一位独立的顾问。

# 技术编辑简介

**David Franson** 自从 20 世纪 90 年代开始，就成为网络、编程、2D 和 3D 计算机图形领域的专家，他编写了 *2D Artwork and 3D Modeling for Game Artists*、*The Dark Side of Game Texturing* 和 *Game Character Design Complete*。

**Don Reamey** 是 TIBCO Software 的构建师/首席工程师，负责 TIBCO Sportfire 商务智能分析软件。在加入 TIBCO 之前，Don 在 Microsoft 做了 12 年的软件开发工程师，其主要工作是 SharePoint、SharePoint Online 和 InfoPath Forms Service。Don 还用 10 年的时间为资本市场编写财务服务软件。

**Mitchel Sellers** 擅长使用 Microsoft 技术进行软件开发。他是 IowaComputerGurus 公司的 CEO，与世界范围内的各种公司一起工作。他是一位 Microsoft C# MVP、Microsoft 认证专家，还是 *Professional DotNetNuke Module Programmig*(Wrox 出版社，2009)的作者。Mitchel 常常在网上撰写技术文章，也经常在用户组和会议上发言。他也是 DotNetNuke 核心团队的一员，.NET 和 DotNetNuke 开发社区的积极参与者。有关 Mitchel 专业经验、认证和出版物的更多信息，可访问 http://mitchelsellers.com/。

来越复杂，既要保证能跟上最新硬件的发展步伐，又要与 20 世纪 90 年代初开始流行的 Windows 产品向后兼容。如果要得到一系列简单而专业的语言、环境和开发工具，让开发人员轻松地编写一流的软件，就需要一个新的开端。

这就是 C#和.NET 的作用。粗略地说，.NET 是一种在 Windows 平台上编程的架构——一种 API。C#是一种从头开始设计的用于.NET 的语言，它可以利用.NET Framework 及其开发环境中的所有新增功能，以及在最近 25 年来出现的面向对象的编程方法。

在继续介绍前，必须先说明，后向兼容性并没有在这个演化进程中丧失。现有的程序仍可以使用，.NET 也兼容现有的软件。现在，在 Windows 上软件组件之间的通信几乎都使用 COM 实现。因此，.NET 能够提供现有 COM 组件的包装器(wrapper)，以便.NET 组件与之通信。

我们不需要学习了 C#才能给.NET 编写代码，因为 Microsoft 已经扩展了 C++，还对 Visual Baisc 进行了很多改进，把它转变成了功能更强大的语言，并允许把用这些语言编写的代码用于.NET 环境。但其他这些语言都因有多年演化的遗留痕迹，并非一开始就用现在的技术来编写，导致它们不能用于.NET 环境。

本书将介绍 C#编程技术，同时提供.NET 体系结构工作原理的必要背景知识。我们不仅会介绍 C#语言的基础，还会给出使用各种相关技术的应用程序对应的示例，包括数据库访问、动态的 Web 页面、高级的图形和目录访问等。

Windows API 自从 1993 年发布的 Windows NT 以来一直在演化和扩展，但自从 2002 年以来，.NET Framework 对程序编写方式进行了重大的修改，2012 年又进行了一次很大的改动。每 10 年就会发生这种改变吗？Windows 8 现在提供了一种新的 API：用于 Windows Store 应用程序的 Windows 运行库(WinRT)。这个运行库是一个本机 API(类似于 Windows API)，它没有把.NET 运行库作为其核心，但提供了基于.NET 理念的非常好的新功能。Windows 8 包含这个 API 的第一个版本，可用于现代模式的应用程序。尽管它不基于.NET，但仍可以将.NET 的一个子集应用于 Windows Store 应用程序，用 C#编写该应用程序。这个新的运行库在未来的几年中会演化，随 Windows 的未来版本一起发布。本书也讨论了如何使用 C#和 WinRT 编写 Windows Store 应用程序。

## .NET 的优点

前面阐述了.NET 的优点，但并没有说它会使开发人员的工作更易完成。本节将简要讨论.NET 的改进功能。

- **面向对象编程**：.NET Framework 和 C#从一开始就完全基于面向对象的原则。
- **优秀的设计**：一个基类库，它以一种非常直观的方式设计出来。
- **语言无关性**：在.NET 中，Visual Baisc、C#和托管 C++等语言都可以编译为通用的中间语言(Intermediate Language)。这说明，语言可以用以前没有的方式交互操作。
- **对动态 Web 页面更好的支持**：虽然经典 ASP 具有很大的灵活性，但效率不是很高，这是因为它使用了解释性的脚本语言，且缺乏面向对象的设计，从而导致 ASP 代码比较混乱。.NET 使用 ASP.NET，为 Web 页面提供了一种集成支持。使用 ASP.NET，可以编译页面中的代码，这些代码还可以使用.NET 能识别的高级语言来编写，如 C#或 Visual Basic 2010。.NET 现在还添加了对最新 Web 技术的重要支持，如 Ajax 和 jQuery。
- **高效的数据访问**：一组.NET 组件，统称为 ADO.NET，提供了对关系数据库和各种数据源的高效访问。这些组件也可用于访问文件系统和目录。尤其是，.NET 内置了 XML 支持，可以处理从非 Windows 平台导入或导出的数据。

- **代码共享**：.NET 引入了程序集的概念，替代了传统的 DLL，可以完美无暇地改进代码在应用程序之间的共享方式。程序集是解决版本冲突的正式设备，程序集的不同版本可以并存。
- **增强的安全性**：每个程序集还可以包含内置的安全信息，这些信息可以准确地指出谁或哪种类型的用户或进程可以调用什么类的哪些方法。这样就可以非常准确地控制用户部署的程序集的使用方式。
- **对安装没有任何影响**：有两种类型的程序集，分别是共享程序集和私有程序集。共享程序集是可用于所有软件的公共库，而私有程序集只用于特殊软件。由于私有程序集完全自包含，所以安装过程非常简单。没有注册表项，只需要把相应的文件放在文件系统的相应文件夹中即可。
- **Web 服务的支持**：.NET 完全集成了对开发 Web 服务的支持，用户可以轻松地开发任何类型的应用程序。
- **Visual Studio 2012**：.NET 附带了一个 Visual Studio 2012 开发环境，它同样可以很好地利用 C++、C#、Visual Basic 2012 和 ASP.NET 或 XML 进行编码。Visual Studio 2012 集成了这个 IDE 所有以前版本中的各种语言专用环境中的所有最佳功能。
- **C#**：是使用.NET 的一种面向对象的强大且流行的语言。

第 1 章将详细讨论.NET 体系结构的优点。

## .NET Framework 4.5 中的新增特性

.NET Framework 的第 1 版(1.0 版)在 2002 年发布，赢得了许多人的喝彩。.NET Framework 2.0 在 2005 年发布，是该架构的一个主要版本。2.0 版本的主要新特性是 C#和运行库中对泛型的支持(为泛型修改了 IL 代码)、新类和接口。.NET 3.0 以 2.0 运行库为基础，引入了创建 UI 的新方式(WPF 和 XAML，基于矢量的图形替代了基于像素的图形)和一个新的通信技术(WCF)。.NET 3.5 和 C# 3.0 引入了 LINQ，这是可用于所有数据源的查询语法。.NET Framework 4 是该产品的另一个重要的版本，也引入了运行库的一个新版本 4.0 和 C#的新版本 4.0，提供了动态语言集成和大量用于并行编程的新库。.NET Framework 4.5 基于 4.0 运行库的更新版本，包含了许多重要的新功能。

对于.NET Framework 的每个版本，Microsoft 总是试图确保对已开发出的代码进行尽可能少的不兼容的更改。到目前为止，Microsoft 在这方面做得很成功。

下面将详细描述 C# 2012 和.NET Framework 4.5 中的一些新变化。

### 异步编程

阻塞 UI 对用户并不友好，如果 UI 不响应，用户就会不耐烦。也许读者在 Visual Studio 中也有这种经历。而 Visual Studio 在这方面好了许多，在许多情形下响应得更快。

.NET Framework 总是提供方法的异步调用。但是，使用同步方法比调用其异步变体容易得多。这在 C# 5.0 版本中有了改变。异步编程与编写同步程序一样容易。新的 C#关键字基于自从.NET 4.0 以来就有的.NET 并行库，现在该语言提供了高效功能。

### Windows Store 应用程序和 Windows 运行库

Windows Store 应用程序可以使用 C#、Windows 运行库和.NET Framework 的一个子集编写，Windows 运行库是一个新的本机 API，提供了类似于.NET 的类、方法、属性和事件。使用语言投射功能，改善了.NET 运行库。在.NET 4.5 中，.NET 4.0 运行库进行了就地更新。

### 数据访问的改善

ADO.NET Entity Framework 提供了重要的新功能。其版本从.NET 4.0 的 4.0 改为.NET 4.5 的 5.0。.NET 4.0 发布后，Entity Framework 已经在 4.1、4.2 和 4.3 中接受了更新。现在新功能，例如 Code First、空间类型、使用枚举、表值功能等，都可以使用了。

### WPF 的改善

WPF 进行了改进，以编写 Windows 桌面应用程序。现在可以在非 UI 线程中填充集合，功能区控件现在是架构的一部分，通过事件的弱引用也更容易实现，数据验证可以用 INotifyDataErrorInfo 接口异步完成；实时绘图功能可以方便地动态排序、分组修改了的数据。

### ASP.NET MVC

Visual Studio 2010 包含 ASP.NET MVC 2.0。Visual Studio 2012 发布后，就可以使用 ASP.NET MVC 4.0 了。ASP.NET MVC 提供了许多开发人员期待的、使用模型-视图-控制器来创建 ASP.NET 应用程序的方式。ASP.NET MVC 在开发人员构建的应用程序中提供了可测试性、灵活性和可维护性。ASP.NET MVC 不是 ASP.NET Web 窗体的替代品，而只是构建应用程序的另一种方式。

## C#的优点

C#在某种程度上可以看作是.NET 面向 Windows 环境的一种编程语言。在过去的 15 年中，Microsoft 给 Windows 和 Windows API 添加了许多功能，Visual Baisc 2010 和 C++也进行了许多扩展。虽然 Visual Baisc 和 C++最终已成为非常强大的语言，但这两种语言也存在问题，因为它们保留了原来的一些遗留内容。

对于 Visual Basic 6 及其早期版本，它的主要优点是很容易理解，许多编程工作都很容易完成，从很大程度上对开发人员隐藏了 Windows API 和 COM 组件结构的详细信息。其缺点是因为 Visual Basic 从来没有实现真正意义上的面向对象，所以大型应用程序很难分解和维护。另外，因为 Visual Basic 的语法继承自 BASIC 的早期版本(BASIC 主要是为了让刚入门的程序员更容易理解，而不是为了编写大型商业应用程序)，所以不能真正成为结构良好或面向对象的编程语言。

另一方面，C++基于 ANSI C++语言定义。它与 ANSI 不完全兼容，因为 Microsoft 在 ANSI 定义标准化之前编写其 C++编译器，但它已经相当接近。但是，这导致了两个问题。首先，ANSI C++是在十几年前的技术条件下开发的，因此它不支持现在的概念(如 Unicode 字符串和生成 XML 文档)，某些古老的语法结构是为以前的编译器设计的(如成员函数的声明和定义是分开的)。其次，Microsoft 同时还试图把 C++演变为一种用于在 Windows 上执行高性能任务的语言，为此不得不在语言中添加大量 Microsoft 专用的关键字和各种库。其结果是在 Windows 上，该语言非常杂乱。让 C++开发人员描述字符串有多少种定义就可以证明这一点：char*、LPTSTR、string、CString(MFC 版本)、

CString(WTL 版本)、wchar_t*、OLECHAR*等。

现在进入.NET 时代——一种全新的环境，它对这两种语言都进行了新的扩展。Microsoft 给 C++添加了许多 Microsoft 专用的关键字，并把 Visual Baisc 演变为 Visual Baisc 2012，保留了一些基本的 Visual Baisc 语法，但在设计上完全不同于原始 Visual Basic，从实际应用的角度来看，Visual Baisc 2012 是一种新语言。

在这里，Microsoft 决定给开发人员提供另一个选择——专门用于.NET、具有新起点的一种语言，即 C#。Microsoft 在正式场合把 C#描述为一种简单、现代、面向对象、类型非常安全、派生自 C 和 C++的编程语言。大多数独立的评论员对 C#的描述改为“派生自 C、C++和 Java”。这种描述在技术上非常准确，但没有表达出该语言的真正优点。从语法上看，C#非常类似于 C++和 Java，许多关键字都相同，C#也使用类似于 C++和 Java 的块结构，并用花括号({})来标记代码块，用分号分隔各行语句。对 C#代码的第一印象是它非常类似于 C++或 Java 代码。但在这些表面的类似性后面，C#学习起来要比 C++容易得多，但比 Java 难一些。其设计比其他语言更适合现代开发工具，它同时具有 Visual Basic 的易用性，以及 C++的高性能、低级内存访问。C#包括以下一些功能：

- 完全支持类和面向对象编程，包括接口和实现继承、虚函数和运算符重载。
- 一致且定义完善的基本类型集。
- 对自动生成 XML 文档的内置支持。
- 自动清理动态分配的内存。
- 可以用用户定义的属性来标记类或方法。这可以用于文档，对编译有一定的影响(例如，把方法标记为只在调试版本中编译)。
- 可以完全访问.NET 基类库，并易于访问 Windows API(如果实际需要它，这就不常见)。
- 可以使用指针和直接访问内存，但 C#语言可以在没有它们的条件下访问内存。
- 以 Visual Basic 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，该组件库可以用与 ActiveX 控件(COM 组件)相同的方式由其他代码调用。
- C#可以用于编写 ASP.NET 动态 Web 页面和 XML Web 服务。

应该指出，对于上述大多数功能，Visual Basic 2012 和 Managed C++也具备。事实上，虽然 C#从一开始就使用.NET，但对.NET 功能的支持不仅更完整，而且在比其他语言更合适的语法环境中提供了这些功能。C#语言本身非常类似于 Java，但其中有一些改进，尤其是，Java 并不应用于.NET 环境。

在结束这个主题前，还要指出 C#的两个局限性。一方面是该语言不适用于编写时间急迫或性能非常高的代码，例如一个要占用 1000 或 1050 个机器周期的循环，并在不需要这些资源时，立即清理它们。在这方面，C++可能仍是所有低级语言中的佼佼者。另一方面是 C#缺乏性能极高的应用程序所需要的关键功能，包括能够指定那些保证在代码的特定地方运行的内联函数和析构函数。但这类应用程序非常少。

## 编写和运行 C#代码的环境

.NET Framework 4.5 运行在 Windows Vista/7/8 和服务器操作系统 Windows Server 2008、2008 R2 和 2012 上。要使用.NET 编写代码，需要安装.NET 4.5 SDK。

此外，除非要使用文本编辑器或其他第三方开发环境来编写 C#代码，否则用户几乎肯定也希望使用 Visual Studio 2012。运行托管代码不需要安装完整的 SDK，但需要.NET 运行库。需要把.NET 运行库和代码分布到还没有安装它的客户端上。

## 本书内容

本书首先在第 1 章介绍.NET 的整体体系结构，给出编写托管代码所需要的背景知识，此后本书分几部分介绍 C#语言及其在各个领域中的应用。

**第Ⅰ部分——C#语言**

本部分给出 C#语言的背景知识。尽管这一部分假定读者是有经验的编程人员，但它没有假设读者拥有任何特殊语言的知识。首先介绍 C#的基本语法和数据类型，再介绍 C#的面向对象功能，之后是 C#中的一些高级编程主题。

**第Ⅱ部分——Visual Studio**

本部分介绍全世界 C#开发人员都使用的主要 IDE：Visual Studio 2012。本部分的两章探讨使用工具构建基于.NET Framework 4.5 的应用程序的最佳方式，另外，本部分还讨论项目的部署。

**第Ⅲ部分——基础**

本部分介绍在.NET 环境中编程的规则。特别是安全性、线程、本地化、事务、构建 Windows 服务的方式，以及将自己的库生成为程序集的方式等主题。其中一部分介绍如何使用平台调用和 COM 交互操作功能，与本地代码和程序集进行交互操作。本部分还讨论了 Windows 运行库与.NET 的区别，以及如何编写 Windows 8 模式的程序。

**第Ⅳ部分——数据**

本部分介绍如何使用 ADO.NET 访问数据库，学习 ADO.NET Entity Framework。我们可以使用核心 ADO.NET 获得最佳性能，而使用 ADO.NET Entity Framework 可以方便地把对象映射到关系上。还讨论了现在可以使用的 Model First、Database First 和 Code First 编程模型。我们还详细说明.NET 对 XML 的支持，以及如何使用 LINQ 查询 XML 数据源。

**第Ⅴ部分——显示**

本部分首先阐述如何编写基于 Windows Presentation Foundation 的应用程序，介绍不同的控件类型、样式、资源和数据绑定，以及如何创建固定的和流畅的文档并打印出来。本部分还会介绍如何创建 Windows Store 应用程序，使用图片生成更漂亮的 UI、网格，以及与其他应用程序交互操作的协定。最后讨论 ASP.NET 提供的许多新功能，用 ASP.NET Web 窗体创建 Web 站点、ASP.NET MVC 和动态数据。

**第Ⅵ部分——通信**

这一部分介绍通信，主要论述独立于平台使用 Windows Communication Foundation(WCF)进行通信的服务，以及使用 WCF 数据服务访问数据。通过消息队列，揭示了断开连接的异步通信。本部分还介绍如何利用 Windows Workflow Foundation(WF)和对等网络。

## 如何下载本书的示例代码

在读者学习本书中的示例时，可以手工输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从本书合作站点 http://www.wrox.com/和 http://www.tupwk.com.cn/downpage 上下载。登录到站点 http://www.wrox.com/上，使用 Search 工具或书名列表就可以找到本书。接着单击本书细目页面上的 Download Code 链接，就可以获得所有的源代码。

**注释：**

许多图书的书名都很相似，所以通过 ISBN 查找本书是最简单的，本书的 ISBN 是 978-1-118-31442-5。

在下载了代码后，只需用自己喜欢的解压缩软件对它进行解压缩即可。另外，也可以进入 http://www.wrox.com/dynamic/books/download.aspx 上的 Wrox 代码下载主页，查看本书和其他 Wrox 图书的所有代码。

## 勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但是错误总是难免的，如果你在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将非常感激。通过勘误表，可以让其他读者避免受挫，当然，这还有助于提供更高质量的信息。

要在网站上找到本书的勘误表，可以登录 http://www.wrox.com，通过 Search 工具或书名列表查找本书，然后在本书的细目页面上，单击 Book Errata 链接。在这个页面上可以查看 Wrox 编辑已提交和粘贴的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 www.wrox.com/misc-pages/booklist.shtml。

如果在 Book Errata 页面上没有看到你找出的错误，请进入 www.worx.com/contact/techsupport.shtml，填写表单，发电子邮件，我们就会检查你的信息，如果是正确的，就在本书的勘误表中粘贴一个消息，我们将在本书的后续版本中采用。

## p2p.wrox.com

P2P 邮件列表是为作者和读者之间的讨论而建立的。读者可以在 p2p.wrox.com 上加入 P2P 论坛。该论坛是一个基于 Web 的系统，用于传送与 Wrox 图书相关的信息和相关技术，与其他读者和技术用户交流。该论坛提供了订阅功能，当论坛上有新帖子时，会给你发送你选择的主题。Wrox 作者、编辑和其他业界专家和读者都会在这个论坛上进行讨论。

在 http://p2p.wrox.com 上有许多不同的论坛，帮助读者阅读本书，在读者开发自己的应用程序时，也可以从这个论坛中获益。要加入这个论坛，必须执行下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读其内容，单击 Agree 按钮。

(3) 提供加入论坛所需的信息及愿意提供的可选信息，单击 Submit 按钮。

(4) 然后就可以收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

**提示：**

不加入 P2P 也可以阅读论坛上的信息，但只有加入论坛后，才能发送自己的信息。

加入论坛后，就可以发送新信息，回应其他用户的帖子。可以随时在 Web 上阅读信息。如果希望某个论坛给自己发送新信息，可以在论坛列表中单击该论坛对应的 Subscribe to this Forum 图标。

对于如何使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作原理，以及许多针对 P2P 和 Wrox 图书的常见问题解答。要阅读 FAQ，可以单击任意 P2P 页面上的 FAQ 链接。

# 目 录

## 第Ⅰ部分 C# 语 言


第 1 章 .NET 体系结构 ····· 2
1.1 C#与.NET 的关系 ····· 2
1.2 公共语言运行库 ····· 3
1.2.1 平台无关性 ····· 3
1.2.2 提高性能 ····· 3
1.2.3 语言的互操作性 ····· 4
1.3 中间语言 ····· 5
1.3.1 面向对象和接口的支持 ····· 6
1.3.2 不同的值类型和引用类型 ····· 6
1.3.3 强数据类型化 ····· 7
1.3.4 通过异常处理错误 ····· 11
1.3.5 特性的使用 ····· 12
1.4 程序集 ····· 12
1.4.1 私有程序集 ····· 13
1.4.2 共享程序集 ····· 13
1.4.3 反射 ····· 14
1.4.4 并行编程 ····· 14
1.4.5 异步编程 ····· 14
1.5 .NET Framework 类 ····· 14
1.6 名称空间 ····· 15
1.7 用 C#创建.NET 应用程序 ····· 16
1.7.1 创建 ASP.NET 应用程序 ····· 16
1.7.2 使用 WPF ····· 18
1.7.3 Windows 8 应用程序 ····· 18
1.7.4 Windows 服务 ····· 18
1.7.5 WCF ····· 19
1.7.6 Windows WF ····· 19
1.8 C#在.NET 企业体系结构中的作用 ····· 19
1.9 小结 ····· 20

第 2 章 核心 C# ····· 22
2.1 C#基础 ····· 23
2.2 第一个 C#程序 ····· 23
2.2.1 代码 ····· 23
2.2.2 编译并运行程序 ····· 23
2.2.3 详细介绍 ····· 24
2.3 变量 ····· 26
2.3.1 变量的初始化 ····· 26
2.3.2 类型推断 ····· 27
2.3.3 变量的作用域 ····· 28
2.3.4 常量 ····· 30
2.4 预定义数据类型 ····· 31
2.4.1 值类型和引用类型 ····· 31
2.4.2 CTS 类型 ····· 32
2.4.3 预定义的值类型 ····· 33
2.4.4 预定义的引用类型 ····· 35
2.5 流控制 ····· 37
2.5.1 条件语句 ····· 37
2.5.2 循环 ····· 41
2.5.3 跳转语句 ····· 44
2.6 枚举 ····· 45
2.7 名称空间 ····· 46
2.7.1 using 语句 ····· 47
2.7.2 名称空间的别名 ····· 48
2.8 Main()方法 ····· 49
2.8.1 多个 Main()方法 ····· 49
2.8.2 给 Main()方法传递参数 ····· 50
2.9 有关编译 C#文件的更多内容 ····· 51
2.10 控制台 I/O ····· 53
2.11 使用注释 ····· 54
2.11.1 源文件中的内部注释 ····· 54
2.11.2 XML 文档 ····· 55

2.12 C#预处理器指令……57
2.12.1 #define 和 #undef……57
2.12.2 #if、#elif、#else 和#endif……58
2.12.3 #warning 和 #error……59
2.12.4 #region 和#endregion……59
2.12.5 #line……60
2.12.6 #pragma……60
2.13 C#编程规则……60
2.13.1 关于标识符的规则……60
2.13.2 用法约定……61
2.14 小结……67
第 3 章 对象和类型……68
3.1 创建及使用类……68
3.2 类和结构……69
3.3 类……69
3.3.1 数据成员……70
3.3.2 函数成员……70
3.3.3 只读字段……82
3.4 匿名类型……83
3.5 结构……84
3.5.1 结构是值类型……85
3.5.2 结构和继承……86
3.5.3 结构的构造函数……86
3.6 弱引用……86
3.7 部分类……88
3.8 静态类……89
3.9 Object 类……89
3.9.1 System.Object()方法……90
3.9.2 ToString()方法……91
3.10 扩展方法……92
3.11 小结……93
第 4 章 继承……94
4.1 继承……94
4.2 继承的类型……94
4.2.1 实现继承和接口继承……94
4.2.2 多重继承……95
4.2.3 结构和类……95
4.3 实现继承……95
4.3.1 虚方法……96
4.3.2 隐藏方法……97
4.3.3 调用函数的基类版本……98
4.3.4 抽象类和抽象函数……99
4.3.5 密封类和密封方法……99
4.3.6 派生类的构造函数……100
4.4 修饰符……105
4.4.1 可见性修饰符……105
4.4.2 其他修饰符……106
4.5 接口……106
4.5.1 定义和实现接口……107
4.5.2 派生的接口……110
4.6 小结……112
第 5 章 泛型……113
5.1 泛型概述……113
5.1.1 性能……114
5.1.2 类型安全……115
5.1.3 二进制代码的重用……115
5.1.4 代码的扩展……116
5.1.5 命名约定……116
5.2 创建泛型类……116
5.3 泛型类的功能……120
5.3.1 默认值……121
5.3.2 约束……122
5.3.3 继承……124
5.3.4 静态成员……125
5.4 泛型接口……125
5.4.1 协变和抗变……126
5.4.2 泛型接口的协变……127
5.4.3 泛型接口的抗变……128
5.5 泛型结构……129
5.6 泛型方法……132
5.6.1 泛型方法示例……132
5.6.2 带约束的泛型方法……133
5.6.3 带委托的泛型方法……134
5.6.4 泛型方法规范……135
5.7 小结……136
第 6 章 数组……137
6.1 同一类型和不同类型的多个对象……137

6.2 简单数组 ····· 138
6.2.1 数组的声明 ····· 138
6.2.2 数组的初始化 ····· 138
6.2.3 访问数组元素 ····· 139
6.2.4 使用引用类型 ····· 140
6.3 多维数组 ····· 141
6.4 锯齿数组 ····· 142
6.5 Array 类 ····· 143
6.5.1 创建数组 ····· 143
6.5.2 复制数组 ····· 144
6.5.3 排序 ····· 145
6.6 数组作为参数 ····· 148
6.6.1 数组协变 ····· 149
6.6.2 ArraySegment<T> ····· 149
6.7 枚举 ····· 150
6.7.1 IEnumerator 接口 ····· 150
6.7.2 foreach 语句 ····· 151
6.7.3 yield 语句 ····· 151
6.8 元组 ····· 156
6.9 结构比较 ····· 157
6.10 小结 ····· 160

第 7 章 运算符和类型强制转换 ····· 161
7.1 运算符和类型转换 ····· 161
7.2 运算符 ····· 161
7.2.1 运算符的简化操作 ····· 163
7.2.2 运算符的优先级 ····· 167
7.3 类型的安全性 ····· 168
7.3.1 类型转换 ····· 168
7.3.2 装箱和拆箱 ····· 172
7.4 比较对象的相等性 ····· 172
7.4.1 比较引用类型的相等性 ····· 172
7.4.2 比较值类型的相等性 ····· 173
7.5 运算符重载 ····· 174
7.5.1 运算符的工作方式 ····· 175
7.5.2 运算符重载的示例：Vector 结构 ····· 176
7.6 用户定义的类型强制转换 ····· 182
7.6.1 实现用户定义的类型强制转换 ····· 184
7.6.2 多重类型强制转换 ····· 189
7.7 小结 ····· 193

第 8 章 委托、Lambda 表达式和事件 ····· 194
8.1 引用方法 ····· 194
8.2 委托 ····· 195
8.2.1 声明委托 ····· 195
8.2.2 使用委托 ····· 196
8.2.3 简单的委托示例 ····· 199
8.2.4 Action<T>和 Func<T>委托 ····· 201
8.2.5 BubbleSorter 示例 ····· 202
8.2.6 多播委托 ····· 204
8.2.7 匿名方法 ····· 208
8.3 Lambda 表达式 ····· 209
8.3.1 参数 ····· 209
8.3.2 多行代码 ····· 210
8.3.3 闭包 ····· 210
8.3.4 使用 foreach 语句的闭包 ····· 211
8.4 事件 ····· 212
8.4.1 事件发布程序 ····· 212
8.4.2 事件侦听器 ····· 214
8.4.3 弱事件 ····· 215
8.5 小结 ····· 219

第 9 章 字符串和正则表达式 ····· 220
9.1 System.String 类 ····· 221
9.1.1 创建字符串 ····· 222
9.1.2 StringBuilder 成员 ····· 225
9.1.3 格式字符串 ····· 226
9.2 正则表达式 ····· 231
9.2.1 正则表达式概述 ····· 231
9.2.2 RegularExpressionsPlayaround 示例 ····· 232
9.2.3 显示结果 ····· 235
9.2.4 匹配、组合和捕获 ····· 236
9.3 小结 ····· 238

第 10 章 集合 ····· 239
10.1 概述 ····· 239

10.2 集合接口和类型 ····· 240
10.3 列表 ····· 241
10.3.1 创建列表 ····· 242
10.3.2 只读集合 ····· 251
10.4 队列 ····· 251
10.5 栈 ····· 255
10.6 链表 ····· 256
10.7 有序列表 ····· 262
10.8 字典 ····· 263
10.8.1 键的类型 ····· 264
10.8.2 字典示例 ····· 265
10.8.3 Lookup 类 ····· 269
10.8.4 有序字典 ····· 270
10.9 集 ····· 270
10.10 可观察的集合 ····· 272
10.11 位数组 ····· 273
10.11.1 BitArray 类 ····· 274
10.11.2 BitVector32 结构 ····· 276
10.12 并发集合 ····· 278
10.12.1 创建管道 ····· 279
10.12.2 使用 BlockingCollection ····· 282
10.12.3 使用 ConcurrentDictionary ····· 283
10.12.4 完成管道 ····· 285
10.13 性能 ····· 286
10.14 小结 ····· 288

第 11 章 LINQ ····· 289
11.1 LINQ 概述 ····· 289
11.1.1 列表和实体 ····· 289
11.1.2 LINQ 查询 ····· 293
11.1.3 扩展方法 ····· 294
11.1.4 推迟查询的执行 ····· 295
11.2 标准的查询操作符 ····· 297
11.2.1 筛选 ····· 299
11.2.2 用索引筛选 ····· 299
11.2.3 类型筛选 ····· 300
11.2.4 复合的 from 子句 ····· 300
11.2.5 排序 ····· 301
11.2.6 分组 ····· 302
11.2.7 对嵌套的对象分组 ····· 303
11.2.8 内连接 ····· 304
11.2.9 左外连接 ····· 305
11.2.10 组连接 ····· 306
11.2.11 集合操作 ····· 309
11.2.12 合并 ····· 310
11.2.13 分区 ····· 311
11.2.14 聚合操作符 ····· 312
11.2.15 转换操作符 ····· 314
11.2.16 生成操作符 ····· 315
11.3 并行 LINQ ····· 316
11.3.1 并行查询 ····· 316
11.3.2 分区器 ····· 317
11.3.3 取消 ····· 317
11.4 表达式树 ····· 318
11.5 LINQ 提供程序 ····· 320
11.6 小结 ····· 321

第 12 章 动态语言扩展 ····· 322
12.1 DLR ····· 322
12.2 dynamic 类型 ····· 323
12.3 包含 DLR ScriptRuntime ····· 327
12.4 DynamicObject 和 ExpandoObject ····· 330
12.4.1 DynamicObject ····· 330
12.4.2 ExpandoObject ····· 332
12.5 小结 ····· 333

第 13 章 异步编程 ····· 334
13.1 异步编程的重要性 ····· 334
13.2 异步模式 ····· 335
13.2.1 同步调用 ····· 342
13.2.2 异步模式 ····· 343
13.2.3 基于事件的异步模式 ····· 344
13.2.4 基于任务的异步模式 ····· 345
13.3 异步编程的基础 ····· 347
13.3.1 创建任务 ····· 347
13.3.2 调用异步方法 ····· 348
13.3.3 延续任务 ····· 348

13.3.4 同步上下文 …… 349
13.3.5 使用多个异步方法 …… 349
13.3.6 转换异步模式 …… 350
13.4 错误处理 …… 351
13.4.1 异步方法的异常处理 …… 352
13.4.2 多个异步方法的异常处理 …… 352
13.4.3 AggregateException 类 …… 353
13.5 取消 …… 353
13.5.1 开始取消任务 …… 354
13.5.2 使用框架特性取消任务 …… 354
13.5.3 取消自定义任务 …… 355
13.6 小结 …… 355

第 14 章 内存管理和指针 …… 356
14.1 内存管理 …… 356
14.2 后台内存管理 …… 356
14.2.1 值数据类型 …… 357
14.2.2 引用数据类型 …… 358
14.2.3 垃圾回收 …… 359
14.3 释放非托管的资源 …… 361
14.3.1 析构函数 …… 361
14.3.2 IDisposable 接口 …… 362
14.3.3 实现 IDisposable 接口和析构函数 …… 363
14.4 不安全的代码 …… 365
14.4.1 用指针直接访问内存 …… 365
14.4.2 指针示例：PointerPlayground …… 374
14.4.3 使用指针优化性能 …… 378
14.5 小结 …… 381

第 15 章 反射 …… 382
15.1 在运行期间处理和检查代码 …… 382
15.2 自定义特性 …… 383
15.2.1 编写自定义特性 …… 383
15.2.2 自定义特性示例：WhatsNewAttributes …… 386
15.3 反射 …… 389
15.3.1 System.Type 类 …… 389
15.3.2 TypeView 示例 …… 392
15.3.3 Assembly 类 …… 393
15.3.4 完成 WhatsNewAttributes 示例 …… 395
15.4 小结 …… 398

第 16 章 错误和异常 …… 399
16.1 简介 …… 399
16.2 异常类 …… 400
16.3 捕获异常 …… 401
16.3.1 实现多个 catch 块 …… 403
16.3.2 在其他代码中捕获异常 …… 407
16.3.3 System.Exception 属性 …… 407
16.3.4 没有处理异常时所发生的情况 …… 408
16.3.5 嵌套的 try 块 …… 408
16.4 用户定义的异常类 …… 410
16.4.1 捕获用户定义的异常 …… 411
16.4.2 抛出用户定义的异常 …… 412
16.4.3 定义用户定义的异常类 …… 415
16.5 调用者信息 …… 417
16.6 小结 …… 418

第Ⅱ部分 Visual Studio

第 17 章 Visual Studio 2012 …… 420
17.1 用 Visual Studio 2012 进行工作 …… 420
17.1.1 项目文件的改进 …… 423
17.1.2 Visual Studio 的版本 …… 423
17.1.3 Visual Studio 设置 …… 424
17.2 创建项目 …… 424
17.2.1 面向多个版本的 .NET Framework …… 425
17.2.2 选择项目类型 …… 427
17.3 浏览并编写项目 …… 430
17.3.1 Solution Explorer …… 430
17.3.2 用代码编辑器进行工作 …… 436
17.3.3 学习和理解其他窗口 …… 438
17.3.4 排列窗口 …… 442

17.4 构建项目 ····· 442
17.4.1 构建、编译和生成 ····· 442
17.4.2 调试版本和发布版本 ····· 443
17.4.3 选择配置 ····· 445
17.4.4 编辑配置 ····· 445
17.5 调试代码 ····· 446
17.5.1 设置断点 ····· 446
17.5.2 使用数据提示和调试器可视化工具 ····· 447
17.5.3 监视和修改变量 ····· 448
17.5.4 异常 ····· 449
17.5.5 多线程 ····· 450
17.5.6 IntelliTrace ····· 451
17.6 重构工具 ····· 451
17.7 体系结构工具 ····· 453
17.7.1 依赖项关系图 ····· 453
17.7.2 层关系图 ····· 454
17.8 分析应用程序 ····· 456
17.8.1 序列图 ····· 456
17.8.2 探查器 ····· 457
17.8.3 Concurrency Visualizer ····· 459
17.8.4 Code Analysis ····· 459
17.8.5 Code Metrics ····· 460
17.9 单元测试 ····· 461
17.9.1 创建单元测试 ····· 461
17.9.2 运行单元测试 ····· 462
17.9.3 预期异常 ····· 463
17.9.4 测试全部代码路径 ····· 463
17.9.5 外部依赖 ····· 464
17.9.6 Fakes Framework ····· 467
17.10 Windows 8、WCF、WF 等 ····· 468
17.10.1 使用 Visual Studio 2012 生成 WCF 应用程序 ····· 468
17.10.2 使用 Visual Studio 2012 生成 WF 应用程序 ····· 470
17.10.3 使用 Visual Studio 2012 生成 Windows Store 应用程序 ····· 470
17.11 小结 ····· 472

第 18 章 部署 ····· 473
18.1 部署是应用程序生命周期的一部分 ····· 473
18.2 部署的规划 ····· 473
18.2.1 部署选项 ····· 474
18.2.2 部署要求 ····· 474
18.2.3 部署.NET 运行库 ····· 475
18.3 传统的部署选项 ····· 475
18.3.1 xcopy 部署 ····· 476
18.3.2 xcopy 和 Web 应用程序 ····· 476
18.3.3 Windows Installer ····· 476
18.4 ClickOnce ····· 477
18.4.1 ClickOnce 操作 ····· 477
18.4.2 发布 ClickOnce 应用程序 ····· 477
18.4.3 ClickOnce 设置 ····· 479
18.4.4 ClickOnce 文件的应用程序缓存 ····· 481
18.4.5 应用程序的安装 ····· 481
18.4.6 ClickOnce 部署 API ····· 482
18.5 Web 部署 ····· 483
18.5.1 Web 应用程序 ····· 483
18.5.2 配置文件 ····· 483
18.5.3 创建 Web Deploy 包 ····· 484
18.6 Windows 8 应用程序 ····· 485
18.6.1 创建应用程序包 ····· 486
18.6.2 Windows App Certification Kit ····· 487
18.6.3 旁加载 ····· 487
18.6.4 Windows 部署 API ····· 488
18.7 小结 ····· 490

第III部分 基 础

第 19 章 程序集 ····· 492
19.1 程序集的含义 ····· 492
19.1.1 程序集的功能 ····· 493
19.1.2 程序集的结构 ····· 493
19.1.3 程序集清单 ····· 494
19.1.4 名称空间、程序集和组件 ····· 494