C# 开发快速入门

[美] Mark Michaelis
Philip Spokas 著

天宏工作室 译

清华大学出版社

(京) 新登字 158 号

C# 开发快速入门

Mark Michaelis & Philip Spokas: C# Developer s Headstart

EISBN: 0-07-219116-3

图书在版编目 (CIP) 数据

C# 开发快速入门/ (美) 米利斯, (美) 斯博卡斯著; 天宏工作室译 . —北京: 清华大学出版社, 2002

书名原文: C# Developer 's Headstart

ISBN 7-302-05783-4

Ⅰ. C… Ⅱ. ①米… ②斯… ③天… Ⅲ. C 语言- 程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字 (2002) 第 060129 号

出 版 者: 清华大学出版社 (北京清华大学学研大厦, 邮编 100084)

http://www.tup.tsinghua.edu.cn

责任编辑: 汤斌浩

封面设计: 常雪影

印 刷 者: 清华大学印刷厂

发 行 者: 新华书店总店北京发行所

开 本: 787× 960 1/16 印张: 12.75 字数: 251 千字

版 次: 2002 年 9 月第 1 版 2002 年 9 月第 1 次印刷

书 号: ISBN 7-302-05783-4/TP · 3421

印 数: 0001～5000

定 价: 23.00 元

作者简介

Mark Michaelis 居住在伊利诺斯州的 Glen Ellyn，他目前是 Real World Technology 公司的高级软件结构设计师，Real World Technology 是一家专门为制造业开发软件的公司。Mark 在伊利诺斯州立大学获得了哲学学士学位，并从伊利诺斯科技大学获得了计算机科学的硕士学位。Mark 还是 Microsoft Certified Solutions Developer（MCSD）以及 Visual Studio Enterprise 产品组的 Microsoft Most Valuable Professional。他是《COM+ Programming from the Ground Up》一书的作者。您可以使用 e-mail 地址 mark_michaelis@dotnetprogramming.com 与 Mark 联系。

Philip Spokas 在 Lisle，IL 的 Benedictine 大学的数学与计算机科学系获得了理学学士学位，他是一位 Microsoft Certified Professional。Philip 具有各方面的开发经验，包括自 1.0 起的所有 Windows 版本、OS/2 以及 UNIX。他是 Real World Technology 公司负责软件开发的副总裁。

致谢

没有哪一本书能由作者一个人出版，许多人的参与才使本书的出版成为可能。

除了阅读每一章并检查技术准确性之外，技术编辑 Bill Burris 还帮助测试了代码，以确保代码的准确性。Marcia Baker 提供了编辑方面的帮助，这是我们非常倾慕的一项任务，因为我们不仅缺乏这方面的专业知识，而且我们甚至缺乏尝试去做的耐心。Timothy Madrid 和 Monika Faltiss 非常高明地在从技术编辑直至出版的过程中将稿件传递到合适的位置。特别感谢 Monika，感谢她在校样阶段花费的大量时间，因为校对任务极为繁重。我们还非常感谢 Ann Sellers 为了使出版本书成为可能所付出的努力。她的任务是非常困难的，她整理了大量其他出版物，而这本来并不在她的职责范围之内。而且对于接受 Mark 独特的评论和观点，她也始终非常积极主动。

我们为之工作的公司 Real World Technology 非常大度地允许我们花时间撰写本书。毫无疑问，离开办公室，我们任何人都无法完成这项工作。因为这段时间与公司产品的另一个版本的发布时间重合，所以这显得尤为可贵。

最后，非常感谢我们的妻子 Elisabeth 和 Cathy。撰写一本书 (即使只有 6 章) 需要花费很多时间，我们清楚地知道，在完成所需的工作时，世界并不会因此而停滞。Elisabeth 和 Cathy 的付出超出了我们的想像。当我们在清晨、深夜、周末甚至在家庭假期努力工作时，她们非常耐心地支持我们。我们真心感谢这两位妻子，她们乐于奉献自己。我们还要感谢孩子们，他们放弃了与爸爸在一起的时间，以便爸爸能够撰写这本书。

简介

虽然 C# 是在不到一年前才向公众宣布的，但是已经有了一些介绍 C# 的书籍，并且无疑将来还会继续出现这样的书籍。在考虑到这一点之后，《C# 开发快速入门》立图与众不同。我们没有撰写另外一本几乎专门介绍语法的书籍，本书使用 60 页介绍了语法，然后转移到一些与 C# 有关的更复杂的问题。通过利用已有的编程知识，读者将能够在阅读本书之后立即开始实际的 C# 编程。

必须具有的编程技术

因为本书包括了 C# 语言的完整介绍，所以并不需要读者在开始阅读本书之前事先了解这种新语言。不过，本书的目标是使开发人员直接开始 C# 编程，而不是纠缠于基本的语法。考虑到这一点，我们希望本书的读者是至少具有另外一种语言的开发经验的程序员。有了这种经验，本书的读者就会发现他们可以在一章的篇幅中迅速掌握 C# 的基本特征，然后开始学习这种语言的一些更困难的领域。

需要的软件

要想尝试本书中的所有实例，您需要安装 .NET Framework SDK Beta 2 或更高版本。这是随 Visual Studio.NET 自动安装的，但您也可以从 Microsoft 的 Web 站点单独下载并安装这个软件。作者推荐安装 Windows 2000 或更高版本，但是 Windows 9x/Windows ME 以及 Windows NT 操作系统也支持 .NET Framework。

不要忘记：Web 上的代码

记住，您可以从 Web 上免费获得本书中所有程序的源代码，地址为 http://www.osborne.com。下载这些代码可使您不必自己键入示例。

快速目录

目录

第一章

C# 简介

本章内容:

基于组件的软件设计模型

基于组件的软件开发的 .NET 方法

什么是 .NET Framework?

为什么使用 C# ?

什么是 C# ?

C# (发音为 C Sharp) 是由 Microsoft 为新的 .NET Framework (.NET 框架) 开发的一种全新的语言。.NET Framework 是 Microsoft 继用于创建基于组件的软件方案的 Windows DNA 之后的下一个开发平台。.NET Framework 反映了 Microsoft 的巨大努力，当然也会影响在今后几年内为 Microsoft 平台开发软件的大多数方式。作为一位开发人员或者开发部经理，您需要了解这项技术，并且越快了解越好。本书的目标是使您领先使用 C# 语言，并介绍运行 C# 的 .NET Framework 的重要特性。

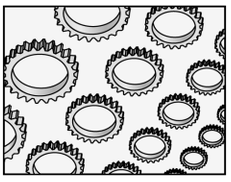
我们希望在您阅读完本书时，您将对 C# 有了基本的理解，并且理解 C# 在 .NET Framework 中的工作方式。没有哪一种语言或者平台能够做所有事情，因此我们还希望能够使您了解 C# 语言目前的一些缺点，这样您就可以事先对此做好准备。

在开始之前需要注意一点：我们尽可能使书中的定义清晰而简洁，这样您就能了解我们所讨论内容的确切含义。全世界的软件公司的销售部门都很含糊地宣称 .NET 这个术语代表了一切，从操作系统、后端服务器平台 (.NET Server 系列) 以及基于 Web 的服务，如 Microsoft 的 Hailstorm 创新，直到近 5 年内在软件产品中置入的所有新特性。不过，在本书中使用 .NET 应该是非常清楚的。在提到术语 .NET 或者 .NET Framework 时，我们是指 Microsoft 通过 .NET Framework SDK 提供的技术。对于 Visual Studio.NET (或者 VS.NET) 所特有的特性或功能，我们将明确指出来。

1.1 基于组件的软件设计模型

在详细讨论 .NET 之前，考虑创建 .NET 的一些基本原因是有益的。现在的 .NET 技术实际上经历了三个重要阶段。第一个重要阶段发生在 90 年代初期，Microsoft 开发了一项名为对象链接与嵌入 (Object Linking and Embedding，OLE) 的技术，用于 OLE 自动化。这项技术是由于当时不同的 Microsoft Office 应用程序进行互操作的需要而产生的。Microsoft 意识到像 Microsoft Word 和 Microsoft Excel 这样的产品各自都是非常优秀的，但是有时用户也希望 Word 文档里面包含电子表格。对此，Microsoft 并不是将所有代码从 Microsoft Excel 复制到 Microsoft Word 中，而是开发了 OLE，它允许将一个程序——例如 Microsoft Excel——中的文档嵌入到一种不同类型的文档 (例如 Microsoft Word) 中。表面上，用户可以将文档从一个应用程序插入到另一个应用程序中。但是在这种表象下面，Microsoft 已经开发了一项重要技术，允许一个应用程序与另一个应用程序通信。不过，应用程序之间的通信并不局限于将一个文档嵌入到另一个文档中。通过 OLE 自动化，一个

应用程序可以调用一个不同的应用程序，这样第一个应用程序就可以操作另一个程序中的对象。



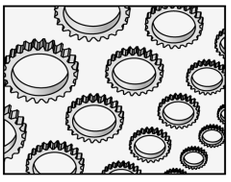
注意

第一代 .NET 的一个主要目标是使进程间通信变得相对容易。特别地，OLE 提供了一种标准的方法来将一种文档嵌入另一种不同的文档中，并允许一个应用程序操作不同应用程序中的对象。

.NET 的下一个重要发展阶段发生于 COM 的引入。随着程序功能的增加，程序变得非常庞大而复杂，难以管理。相对于可执行程序中的代码数量，程序复杂性通常会呈指数级增加而不是线性增加。使用整体方法（一个庞大的可执行文件）的开发人员必须非常了解他们自己的代码以及模块中的许多其他代码。在整体方法中，对一部分代码进行很小的更改也会在程序中的其他位置产生重大影响。这不仅增加了开发工作，而且增加了开发优质软件所需的测试工作。

对整体软件的一个解决方案是基于组件的模型。在基于组件的模型中，软件程序被划分为多个组件，每一个组件都提供了一项不同的服务或者一组不同的功能。以这种方式开发的软件有时被称为组件软件（componentware）。例如，与编写一个很大的可执行文件，其中包括了用于压缩文件、发送 e-mail 以及绘制各种趋势图的代码相比，开发人员意识到将程序划分成单独的模块会更好：一个用于发送 e-mail，一个用于绘制图形，一个用于压缩文件。然后，单独的模块被组合起来并向用户展现为一个应用程序，即使该应用程序在内部可能被分成了许多不同的部分（在逻辑上或者在物理位置上）。

在使用基于组件的方法时，每一个模块都是独立的。因此，每一个程序员都只需管理他所开发的那一个组件。因为每一个组件都是独立的，所以每一个组件的内部结构对于另一个组件的内部结构的影响很小，或者根本没有影响。通过组件模型的分治（divide-and-conquer）技术，应用程序的复杂性降低了。每一个应用程序都可以独立于最终将部署它的系统进行测试。



注意

组件模型减少了软件的整体复杂性，并且大大减少了跨多家公司的分布式开发的难度。软件公司可以在他们擅长的领域开发组件，而其他公司将多家公司开发的组件组合到一个功能齐全的软件包中。

基于组件的模型的另一个重要优点是组件开发人员并不需要实现所有功能。如果开发人员想要开发一个用于追踪公司费用的程序，那么这个程序员不必编写 e-mail 程序，以通过这个程序提交费用报表。相反，开发人员将获得一个由另一家公

司编写的 e-mail 组件，并调用那个组件来实现这种功能。用来向用户显示费用的电子表格控件可能也是这样。或许甚至报表模块也可以来自于第三方。基于组件的模型允许进行跨多家公司或者多个部门的分布式开发。通过这种方式，每一家公司都可以在他们所擅长的领域编写功能，并在需要时使用来自其他公司的第三方组件。

Microsoft 的组件模型实现被称为组件对象模型（Component Object Model, COM）。COM 实际上将 OLE 自动化发展到了一个新的水平，将进程间通信的思想扩展到了模块间通信。应用程序不仅能够进行交互，还可以将二进制模块（如 DLL）加载到进程中，并访问 DLL 中的对象。这些 DLL 中的对象被称为 COM 组件，而对象代码所处的模块被称为 COM 服务器。（注意，组件这个术语在 COM 中是很含混的。组件有时是指对象所处的模块——COM 服务器，有时组件就是对象本身，而不管是否实例化了该对象。在本书中，组件是指对象本身。）

基于组件的模型并不是从 COM 开始的。在 COM 出现之前，开发人员使用库模块，这种模块也提供了服务。使用库模块方法的问题在于加载库以及调用库函数的工作量很大，并且易于出现错误。程序经常找不到库，并且如果出现版本不匹配，那么结果通常会非常糟糕。另外，库模块缺少一种标准的方法。内存管理不存在标准的约定，甚至不存在标准的调用约定。而且库模块并不适合面向对象的编程，它更适合过程编程。其区别在于：面向对象的编程通过定义对结构进行操作并由结构拥有的函数来扩展基本的数据结构。库模块通常只传递数据结构并调用函数，而不是传递对象并在这些对象上调用函数。

在第一次发布 COM 后，又出现了一项名为 ActiveX 的技术。ActiveX 控件是一些组件，专门设计成被包括在其他窗口或者窗体（称为 COM 容器）中。虽然这并不是革命性的，但是 ActiveX 允许开发人员只实现与他们的组件有关的接口，从而大大减少了创建 COM 控件所需的工作量。在 .NET 的发展历程中，ActiveX 并不是一个新的阶段，而只是 COM 阶段的一个改进。

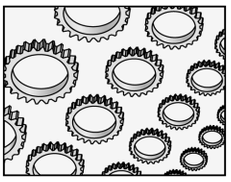
.NET 就是在这种背景下发展出来的。尽管 COM 向软件开发的组件模型提供了巨大好处，它在一些重要的领域仍然很欠缺。

1.2　基于组件的软件开发的 .NET 方法

在开始组件模型的第三个阶段时，我们也开始了一个新的互操作水平。COM 可以在模块之间提供标准的二进制通信机制。而使用 .NET，这一标准就从二进制级别上升到了一种中间语言，这种语言称为 Microsoft IL（MSIL），或者简称为 IL。

注意

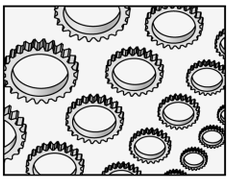
使用 .NET，模块间通信的标准从二进制调用约定和内存管理标准上升到了一种中间



语言，所有 .NET 语言都必须编译为这种语言。

换句话说，.NET 使用一种中间语言代替了 COM 的调用约定和内存管理标准，所有其他 .NET 语言都将编译为这种中间语言。结果就是：不必再由程序员来确保他们的代码将与一种确定的二进制标准进行互操作，而是由不同的 .NET 语言编译器将代码编译到 IL 中来确保这种互操作，而 IL 与其他 IL 模块是自动兼容的。开发人员无需确保他们通过引用计数适当地执行了内存管理，.NET 技术会通过一种机制自动执行内存管理，这种机制通常称为无用单元收集（garbage collection）。我们将在第 3 章中讨论具体的无用单元收集算法的细节。

IL 代码一个重要特征实现了 .NET Framework 中的许多特性，它就是元数据是 IL 代码的一个固有特征。元数据是一种关于数据的数据，它描述了数据的特征。IL 代码中的每一种数据类型定义都随数据类型包括了元数据，即使在代码已经从 .NET 语言（C#、VB 和 Managed C++ 等）编译到了 IL 代码中之后也是如此。



注意

IL 代码的一个重要部分是模块中关于的数据类型的元数据。

元数据并不局限于数据类型本身，.NET 软件模块本身也包括元数据。这种元数据在运行时使用，以确定代码的位置，然后加载那些代码，这样就能执行代码。

1.3 什么是 .NET Framework?

既然您已经了解了 .NET 的起源，下面我们将花更多的时间来定义它。这一节提供了一个简明的定义，帮助您在具体的上下文中理解 .NET。.NET Framework 以及 SDK 本身可以视为三种截然不同的技术：公共语言运行时（Common Language Runtime，CLR）、框架类库（Framework Class Library）以及工具和实用程序。

下面简要说明了每一项技术，这样您就可以了解它们的含义，不过对整个 .NET Framework 的详细说明超出了本书的范围。我们在不同的章节中讨论某些关键领域。例如，我们在第 5 章详细介绍了确定析构（deterministic destruction）。很难一下理解 .NET Framework，但是使用 C# 使得从 .NET 获得好处更为容易，而不需要对它的深入了解。

1.3.1　.NET 的公共语言运行时

CLR 是 .NET Framework 的核心，它负责加载和执行 C# 程序以及使用任何其他 .NET 语言编写的程序。CLR 本身包括在为 .NET 以及 Internet 信息服务(作为 ISAPI 筛选器) 和 Internet Explorer 中的 ASP.NET (以进行客户端编程) 开发的 Windows 可执行程序中。此外，CLR 定义了一些接口，允许自定义的应用程序包括它 (如果现有的选项不合适)。

.NET 在结构上的重大改进并不是 C# 所独享的，但是作为一种 .NET 语言，C# 从这种改进中获得了好处。实际上，C# 的功能主要来自于 .NET Framework 以及 CLR。对 CLR 的主要特性以及优点的简要总结如下所示:

MSIL 的托管执行，包括类型安全性以及无用单元收集的内存管理
丰富的内置类型系统
安全性
与非 .NET 软件的互操作性
改进的部署
公共语言规范 (CLS)

托管的执行

我们在前面讨论了托管的执行的重要方面，但是我们应该说明 IL 是实时 (Just-In-Time) 编译为本地可执行代码的。这会导致在第一次运行程序时的性能降低，但是随后的执行会很快。如果需要，程序集也可以是在运行前编译的，从而消除这种初始的第一次性能降低的现象。

托管的执行环境还定义了应用程序范围，它提供了在程序集之上的执行和安全环境。

类型系统

CLR 提供了一种所有 .NET 语言和类库公用的类型系统。它支持各种类型，包括一组整型及浮点数据类型，以及齐全的字符串类型、日期类型和对象类型。对象类型作为 .NET 中所有其他类型的根类。

因为 .NET 语言不必再尝试定义能够互操作的类型，所以公共类型系统是 .NET 语言互操作特性的关键部分。同时，可以很容易地从 .NET 提供的基本类型构造某种特殊语言所特有的类型。

改进的部署

虽然最近在软件安装和配置方面有了很大的进展，但是应用程序 (无论是“胖”客户还是基于 Web 的应用程序) 的部署仍然是 IT 部门面临的一个开销很大的

问题。对于许多功能,“胖”客户并不会很快消失,其部署方面的进展仍然是非常重要的。此外,随着 Web 服务器变得越来越普及,对于需要部署几十台 Web 服务器或者内联网服务器的 IT 部门而言,部署和维护变得非常重要。即使 Web 客户(特别是包括 ActiveX 下载的那些客户)也会遇到客户的版本控制问题。

.NET 包括了一些进展,它们允许程序集的端到端运行,并促进了批量复制式的部署,在这种部署方式中,文件被复制到目标计算机上而不需要进行注册。.NET 在程序集的主键数据中包括了版本信息,允许在同一台计算机上运行同一个程序集的不同版本。如果一个现有应用程序依赖于某个程序集的特定版本,那么另一个包含同一个程序集的新版本的应用程序不会干扰原来的应用程序。

.NET 还提供了秘密部署程序集或者在全局程序集缓存(Global Assembly Cache,GAC)中部署程序集的选项。

健壮的安全环境

目前软件行业面临的另一个重要问题是安全。猖獗的病毒破坏已经浪费了大量时间和资金。.NET 运行时通过一些重要特性直接处理这个问题以及与安全有关的其他问题。

程序集可以进行数字签名,从而允许对作者进行确认,以避免执行未授权的代码(不管是从命令行执行代码还是当前正在运行的程序调用了它)。

即使您可能值得信任,您也可能会获得并执行不应该信任的代码。当然,执行这些代码并不是您的错,但是总之可能会产生破坏。.NET 运行时安全性提供了一个安全包装器,即使在授权用户执行未授权的代码时,它也可以实施安全措施。

为了提供健壮而安全的环境,.NET 运行时不允许基于指针的访问,除非指针是在一个安全环境中执行的,并且已经特别确认了执行这种“不安全”代码的安全性。这针对有意和无意的灾难提供了另一层保护。

公共语言规范

.NET 公共语言规范(Common Language Specification, CLS)允许 .NET 平台上语言的互操作性,这是通过定义互操作所需的规则来实现的。.NET 平台本身做了大量工作来使之成为可能,但是如果没有明确定义的规则,那么一种语言就不能很轻松地使用在另一种语言中创建的类。

CLS 本身定义了每一种 .NET 语言必须支持的基类型以及应该遵循的约定。最后,CLS 允许独立的第三方为 .NET 创建语言编译器,这些编译器可以具有与 Microsoft 提供的语言相同的互操作特性。