宋智军 邱仲潘 编著

Visual C# 2010 从入门到精通

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

Visual C# 2010从入门到精通

宋智军　邱仲潘　编著

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

内 容 简 介

本书坚持以语言为基础和应用为主导的编写原则，详细介绍如何使用 Visual C# 2010 开发控件台应用程序、Windows 窗体应用程序等。为了更好地帮助读者在短时间内掌握使用 C#语言开发各种应用程序中的知识点和编程技巧，全书的基础知识介绍清晰，理论联系实际，具有很强的操作性。本书还提供了大量的通过测试可运行的完整实例，这些实例都有设计步骤、代码详解、程序运行结果等，不但复习了前面所学的内容，而且还增加了一定的创作技巧。对于容易出现问题的地方，则以“注意”的方式介绍常用的技巧和注意事项。

本书适合学习 Visual C# 2010 的初、中级读者阅读和使用。

图书在版编目（CIP）数据

Visual C# 2010 从入门到精通/宋智军，邱仲潘编著 —北京：电子工业出版社，2011.1
ISBN 978-7-121-12069-5

Ⅰ. ①V… Ⅱ. ①宋… ②邱… Ⅲ. ①C 语言—程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2010）第 205916 号

责任编辑：吴 源
特约编辑：郭 莉
印 刷：北京天竺颖华印刷厂
装 订：三河市鑫金马印装有限公司
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
北京市海淀区翠微东里甲 2 号 邮编：100036
开 本：787×1092 1/16 印张：22.25 字数：569 千字
印 次：2011 年 1 月第 1 次印刷
定 价：42.00 元

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。

前　言

在Visual Studio 6.0之后，微软发布了.NET平台，其后的Visual Studio都是基于.NET平台的，虽然无论是开发环境的改善，还是新的开发语言C#的引入，都没有使得Visual Studio能够再创辉煌，再续经典。每次新版本的发布，程序员们只看到安装程序越来越大，整个开发环境越来越臃肿，却没有看到多少革命性的变化，这多少有点让我们这些天天使用Visual Studio的程序员们有些失望，难道真的是“英雄迟暮”？

直到现在，随着Visual Studio 2010的发布，微软同时宣称会将它打造成与Visual Studio 6.0一样的经典产品，这又让我们就快冰凉的心重燃希望。新版本什么样？有什么新功能？真的会成为下一个经典吗？关心Visual Studio的人都在问，迫切地想先睹为快。现在机会来了，新版本的发布，让我们有机会一睹Visual Studio 2010的芳容，看看微软将如何一步步打造这个未知的“经典”产品。

根据微软发布的一份官方文档宣称，Visual Studio 2010和.NET Framework 4.0将在下面5个方面有所创新：

（1）民主化的应用程序生命周期管理

在一个组织中，应用程序生命周期管理（ALM）将牵涉到多个角色。但是在传统意义上，这一过程中的每个角色并不是完全平等的。Visual Studio Team System 2010将坚持打造一个功能平等、共同分担的平台以用于组织内的应用程序生命周期管理过程。

（2）顺应新的技术潮流

每年，业界内的新技术和新趋势层出不穷。通过Visual Studio 2010，微软将为开发者提供合适的工具和框架，以支持软件开发中最新的架构、开发和部署。

（3）让开发商惊喜

从Visual Studio的第一个版本开始，微软就将提高开发人员的工作效率和灵活性作为自己的目标。Visual Studio 2010将继续关注并且显著地改进开发者最核心的开发体验。

（4）下一代平台浪潮的弄潮儿

微软将继续投资于市场领先的操作系统、工具软件和服务器平台，为客户创造更高的价值。使用Visual Studio 2010，将可以在新一代的应用平台上，为你的客户创造令人惊奇的解决方案。

（5）跨部门的应用

客户将在不同规模的组织内创建应用，跨度从单个部门到整个企业。Visual Studio 2010将确保在这么宽泛的范围内的应用开发都得到支持。

看了这么高度的概括，相信大家都会迷迷糊糊，不知所云。这就是大公司“枪文”的厉害之处：让你听得云里雾里，又让你觉得它的东西很好。实际上，上面这段官方文档可以翻译成程序员更好理解的文字：

“Visual Studio 2010将致力于引领下一代平台技术的发展、提高开发人员的工作效率和热情、创建突破性的应用程序、推动云计算等新兴技术趋势、合理化应 用程序生命周期管理（ALM）。另外，Visual Studio 2010将支持统一建模语言（UML）和特定域语言（DSL），它将可以为各类开发人员提供合适的工具，而这也是微软更大模型平台的一部分。”

这样的介绍是不是更好理解一些呢？上面的介绍都是高度地概括Visual Studio 2010的新特性，实际上，我们希望能够看到一些更加具体、更加实际的内容。所以，在接下来的系列介绍中，会以Visual Studio 2010为基础，带领大家一同来看看Visual Studio 2010中的新内容，从现在开始体验下一代Visual Studio。根据微软提供的一份指导文档为蓝本，本书将依次介绍其全新的基于WPF创建的IDE、增强的代码编辑器、C#中所带来的一些令人激动不已的新特性和Office开发等。

关于本书

本书坚持以语言为基础和应用为主导的编写原则，详细介绍如何使用Visual C# 2010开发控件台应用程序、Windows窗体应用程序和ASP.NET应用程序。为了更好地帮助读者在短时间内掌握使用C#语言开发各种应用程序中的知识点和编程技巧，全书的基础知识介绍清晰，理论联系实际，具有很强的操作性。本书还提供了大量的通过测试可运行的完整实例，这些实例都有设计步骤、代码详解、程序运行结果等，不但复习了前面所学的内容，而且还增加了一定的创作技巧。对于容易出现问题的地方，则以“注意”的方式介绍常用的技巧和注意事项。

本书由宋智军、邱仲潘编著，同时感谢冯姝慧、邹文、邓欣欣、王帅、朱敏、张朋丽等给予的帮助和支持。还要感谢电子工业出版社的编辑和出版，他们为本书的选题策划、编辑加工和出版发行付出了辛勤的劳动。由于编者水平有限，加之时间仓促，书中难免有疏漏和不足之处，恳请专家和广大读者指正。

为方便读者阅读，若需要本书配套资料，请登录“北京美迪亚电子信息有限公司”（http://www.medias.com.cn），在“资料下载”页面进行下载。

目　录

第1章 Visual C# 2010概述

1.1 C#语言和.NET Framework介绍

C#是一种简洁、类型安全的面向对象的语言，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。使用C#可以创建传统的Windows客户端应用程序、XML Web Services、分布式组件、客户端-服务器应用程序、数据库应用程序以及很多其他类型的程序。Microsoft Visual C# 2010提供高级代码编辑器、方便的用户界面设计器、集成调试器和许多其他工具，以在C#语言和.NET Framework的基础上加快应用程序的开发。

1.1.1 C#语言

C#语法表现力强，而且简单易学。C#的大括号语法使任何熟悉C、C++或Java的人都可以立即上手。了解上述任何一种语言的开发人员通常在很短的时间内就可以开始使用C#高效地进行工作。C#语法简化了C++的诸多复杂性，并提供了很多强大的功能，例如可为null的值类型、枚举、委托、lambda表达式和直接内存访问，这些都是Java所不具备的。C#支持泛型方法和类型，从而提供了更出色的类型安全和性能。C#还提供了迭代器，允许集合类的实施者定义自定义的迭代行为，以便容易被客户端代码使用。

作为一种面向对象的语言，C#支持封装、继承和多态性的概念。所有的变量和方法，包括Main方法（应用程序的入口点），都封装在类定义中。类可能直接从一个父类继承，但它可以实现任意数量的接口。重写父类中的虚方法的各种方法要求override关键字作为一种避免意外重定义的方式。在C#中，结构类似于一个轻量类；它是一种堆栈分配的类型，可以实现接口，但不支持继承。除了这些基本的面向对象的原理之外，C#还通过几种创新的语言构造简化了软件组件的开发，这些结构包括：

- 封装的方法签名（称为“委托”），它实现了类型安全的事件通知；
- 属性（Property），充当私有成员变量的访问器；
- 特性（Attribute），也有人称之为属性，提供关于运行时类型的声明性元数据；
- 内联XML文档注释。语言集成查询（LINQ），提供了跨各种数据源的内置查询功能。

在C#中，如果必须与其他Windows软件（如COM对象或本机Win32 DLL）交互，则可以通过一个称为“互操作”的过程来实现。互操作使C#程序能够完成本机C++应用程序可以完成的几乎任何任务。在直接内存访问必不可少的情况下，C#甚至支持指针和“不安全”代码的概念。

C#的生成过程比C和C++简单，比Java更为灵活。没有单独的头文件，也不要求按照特定顺序声明方法和类型。C#源文件可以定义任意数量的类、结构、接口和事件。

1.1.2 .NET Framework平台体系结构

C#程序在.NET Framework上运行，它是Windows的一个不可或缺的组件，包括一个称为

公共语言运行库（CLR）的虚拟执行系统和一组统一的类库。CLR是Microsoft的公共语言基础结构（CLI）的商业实现。CLI是一种国际标准，是用于创建语言和库在其中无缝协同工作的执行和开发环境的基础。

用C#编写的源代码被编译为一种符合CLI规范的中间语言（IL）。IL代码与资源（例如位图和字符串）一起作为一种称为程序集的可执行文件存储在磁盘上，通常具有的扩展名为.exe或.dll。程序集包含清单，它提供有关程序集的类型、版本、区域性和安全要求等信息。

执行C#程序时，程序集将加载到CLR中，这可能会根据清单中的信息执行不同的操作。然后，如果符合安全要求，CLR就会执行实时（JIT）编译以将IL代码转换为本机机器指令。CLR还提供与自动垃圾回收、异常处理和资源管理有关的其他服务。由CLR执行的代码有时称为“托管代码”，它与编译为面向特定系统的本机机器语言的“非托管代码”相对应。图1.1阐释了C#源代码文件、.NET Framework类库、程序集和CLR的编译时与运行时的关系。

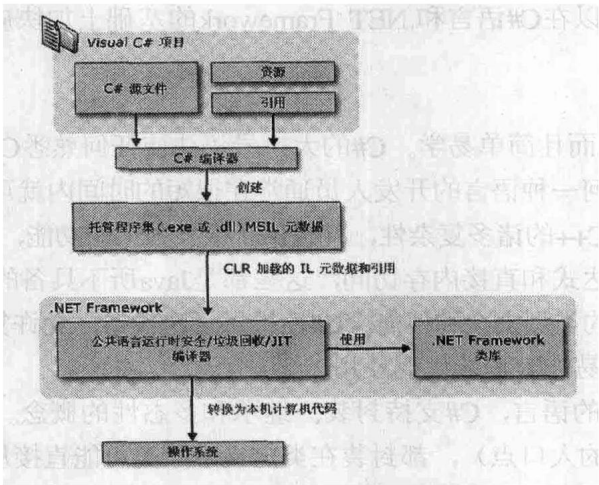

图1.1 C#应用程序的编译与运行流程

语言互操作性是.NET Framework的一项主要功能。因为由C#编译器生成的IL代码符合公共类型规范（CTS），因此从C#生成的IL代码可以与从Visual Basic、Visual C++、Visual J#的.NET版本或者其他20多种符合CTS的语言中的任何一种生成的代码进行交互。单一程序集可能包含用不同.NET语言编写的多个模块，并且类型可以相互引用，就像它们是用同一种语言编写的。

除了运行时服务之外，.NET Framework还包含一个由4000多个类组成的内容详尽的库，这些类被组织为命名空间，为从文件输入和输出、字符串操作、XML分析到Windows窗体控件的所有内容提供了各种有用的功能。典型的C#应用程序使用.NET Framework类库广泛地处理常见的“日常”任务。

1.2 C# 4.0新特性

回顾C#发展的历史，C# 1.0完全是模仿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加入了一堆语法糖，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的原因，如性能之类的而没有采用，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。

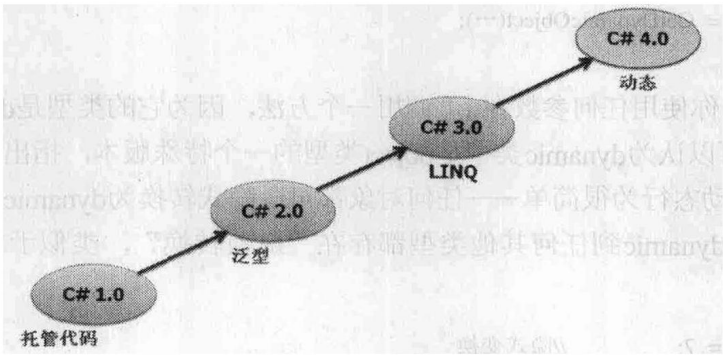

图1.2 C#的演化

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：

（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。

1.2.1 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，运行时会帮你指出针对特定的对象，这些操作的具体意义。这种方法带来了巨大的灵活性，并能最大程度地精简代码，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，则只有到运行时才能得到错误。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。

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。运行时会检查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”，类似于在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
```

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。

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，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。

1.2.2 参数命名和可选参数

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，而无需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。

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)
```

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时只希望忽略y，可以写：

```
M(1, z: 3）;       //通过实参名z进行传递
M(x: 1, z: 3）;    //通过实参名x和z进行传递
M(z: 3, x: 1）;    //颠倒实参顺序
```

这几种形式都是等价的，不过参数总是按照其出现的顺序进行求值，因此对于最后一个示例来说，3会在1之前求值。可选参数和命名参数不仅可以用在方法调用中，还可以用在索引器和构造器中。

重载解析

命名参数和可选参数影响了重载解析，但产生的变化相当简单。如果所有的参数或者是可选的，或者在调用时（通过名字或位置）明确提供了对应的实参，并且实参能够转换为形参类型，则该重载方法是可适用的（applicable）。转换的最优原则只用于明确给定的实参（出于最优的目的），忽略掉的可选参数在重载解析时将不做考虑。如果两个重载方法一样好，则选择没有忽略可选参数的那个。

```
M(string s, int i = 1);
M(object o);
M(int i, string s = “Hello”);
M(int i);
M(5);
```

对于给定的这些重载，我们可以看看上述规则的工作方式。M（string,int）不是可适用的，因为5不能转换为string。M（int,string）是可适用的，因为它的第二个参数是可选的，然后很明显，M（object）和M（int）也是可适用的。M（int,string）和M（int）都比M（object）要好，因为将5转换为int优于将5转换为object。最后，M（int）优于M（int,string），因为它没有被忽略的可选参数。因此，最终调用的方法是M（int）。

1.2.3 COM互操作特性

动态查找以及命名参数和可选参数极大地改善了与COM API（如Office Automation API）互操作的体验。为了减少更多的速度损失，C# 4.0还添加了大量特定于COM的小特性。

动态引入

很多COM方法接受并返回可变类型，这在PIA（Primary Interop Assembly，主互操作程序集）中会表现为object。在绝大多数情况下，程序员在调用这些方法之前就已经从上下文中知道了一个返回值对象的静态类型，但为了使用这些知识，必须明确地在返回值上进行类型转换。这些转换非常普遍，带来了巨大的麻烦。为了得到无缝体验，现在我们可以选择使用dynamic类型来代替可变类型的方式。换句话说，COM方法中出现的是dynamic而不是object。这意味着我们可以直接在返回的对象上访问成员，或者可以使用强类型的局部变量为其赋值，而无需进行转换。例如，我们可以写：

```
excel.Cells[1, 1].Value = "Hello";
Excel.Range range = excel.Cells[1, 1];
```

而不用写：

```
((Excel.Range)excel.Cells[1, 1]).Value = "Hello";
Excel.Range range =(Excel.Range)excel.Cells[1, 1];
```

无PIA的编译

主互操作程序集（Primary Interop Assembly）是从COM接口生成的大型.NET程序集，用于协助完成强类型的互操作。它们为设计提供了巨大的支持，就好像其中的类型真的是用.NET定义的一样。然而，在运行时这些大型程序集很容易使程序膨胀，而且很容易导致版本问题，因为它们是分布式的，不依赖我们的应用程序。PIA特性允许我们在设计时继续使用PIA，而无需在运行时使用它们。C#编译器会将程序中实际用到的PIA中的一小部分直接编译到程序集中。在运行时无需加载PIA。

省略ref

由于采用了不同的编程模型，很多COM API包含大量的引用参数。与C#中的ref相反，这些参数并不意味着要修改传入的实参以供调用方之后使用，而只是另外一种传递参数值的简单方式。C#程序员必须为所有这些ref参数创建临时变量，并按引用进行传递，这看上去一点也不合理。因此，对于COM方法，C#编译器允许按值传递这些参数，并自动生成存放传入值的临时变量，并在调用返回后丢弃这些变量。使用这种方式，调用方看到的语义是按值传递，而且不会有任何副作用，而被调用的方法得到的依然是一个引用。

1.2.4 变性

泛型的某个方面会让人感到奇怪，比如下面的代码是不合法的：

```
IList<string> strings = new List<string>();
IList<object> objects = strings;
```

第二个赋值是不允许的，因为strings和objects的元素类型并不一样，这会破坏类型安全性。然而，对于某些接口来说上述情况并不会发生，尤其是不能将对象插入集合时。例如IEnumerable

<T>就是这样的接口。如果改为：

```
IEnumerable<object> objects = strings;
```

这样就没法通过objects将错误类型的对象插入到strings中了，因为objects没有插入元素的方法。变性（variance）就是用于在这种能保证安全的情况下进行赋值的。

协变性

在.NET 4.0中，IEnumerable<T>接口将会按照下面的方式进行定义：

```
public interface IEnumerable<out T> : IEnumerable
{
        IEnumerator<T> GetEnumerator();
}
public interface IEnumerator<out T> : IEnumerator
{
        bool MoveNext();
        T Current { get; }
}
```

这些声明中的“out”指出T只能出现在接口的输出位置（如果不是这样，编译器会报错）。有了这一限制，接口对于T类型就是“协变的”，这意味着如果A可以按引用转换为B，则IEnumerable<A>可以当做IEnumerable<B>使用。其结果是，任何一个字符串序列也就是一个对象序列了。这很有用，例如在LINQ方法中，使用上面的定义：

```
var result = strings.Union(objects);        //使用了IEnumerable<object>
```

之前这样做是不允许的，我们必须做一些麻烦的包装，使得两个序列具有相同的元素类型。

逆变性

类型参数还可以具有“in”修饰符，限制它们只能出现在输入位置上。例如IComparer<T>：

```
public interface IComparer<in T>
{
        public int Compare(T left, T right);
}
```

其结果有点让人迷惑，就是IComparer<object>可以作为IComparer<string>使用！这样考虑这个结果就会很有意义——如果一个比较器可以比较任意两个object，它当然也可以比较两个string。这种性质被称作“逆变性（contravariance）”。泛型类型可以同时拥有带有in和out修饰符的类型参数，例如Func<...>委托类型：

```
public delegate TResult Func<in TArg, out TResult>(TArg arg);
```

很明显参数永远都是传入的，而结果永远只能是传出的。因此，Func<object, string>可以用作Func<string, object>。

限制

变性类型参数只能在接口和委托类型中声明，这是CLR的限制。变性只能应用在类型参数的引用转换之间。例如，IEnumerable<int>不能作为IEnumerable<object>使用，因为从int到object的转换是装箱转换，而不是引用转换。还要注意的是，CTP中并没有包含前面提到的.NET类型的新版本。为了试验变性，我们需要自己声明变性接口和委托类型。

1.3 Visual Studio 2010的运行环境及安装

在安装Visual Studio 2010版本时，应预先考虑多个相关项，以减少安装过程中遇到问题的可能性。例如，预先确定如何配置目标计算机以及是否安装了其他版本。安装了Visual Studio 2010后，可以执行诸如注册产品和安装可选组件等其他步骤。下面列出了安装Visual Studio 2010前要考虑的硬件和软件环境事项，如表1.1所示。

表1.1 安装环境

安装环境	要求
CPU	最低要求：1.6GHz，建议：2.6GHz
内存	最低要求：512MB，建议：1GB
硬盘	最低要求：3GB，建议：20GB
操作系统	Windows Server 2003/2008，Windows Vista，Windows XP，Windows 7
平台架构	32-Bit，64-Bit
Web服务器	Microsoft因特网信息服务系统（IIS）5.0或以上版本
IE浏览器	Internet Explorer 6或以上版本
运行环境	Microsoft.NET Framework SDK
开发环境	Microsoft Visual Studio.NET

1.3.1 安装Visual Studio 2010

在Windows XP操作系统平台上安装Visual Studio 2010的步骤如下：

第1步：关闭所有打开的应用程序，以免增加计算机在安装期间重新启动的次数。

第2步：放入安装光盘，双击setup.exe打开Visual Studio 2010安装程序启动窗口，如图1.3所示。

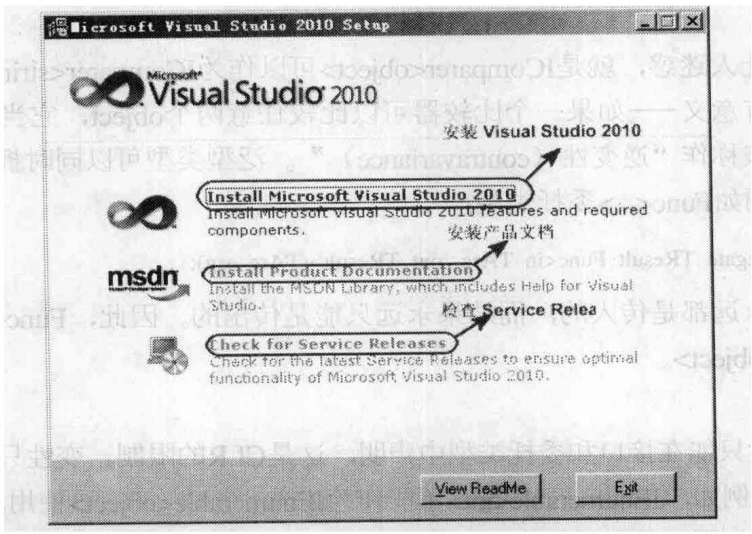

图1.3 Visual Studio 2010安装程序启动界面