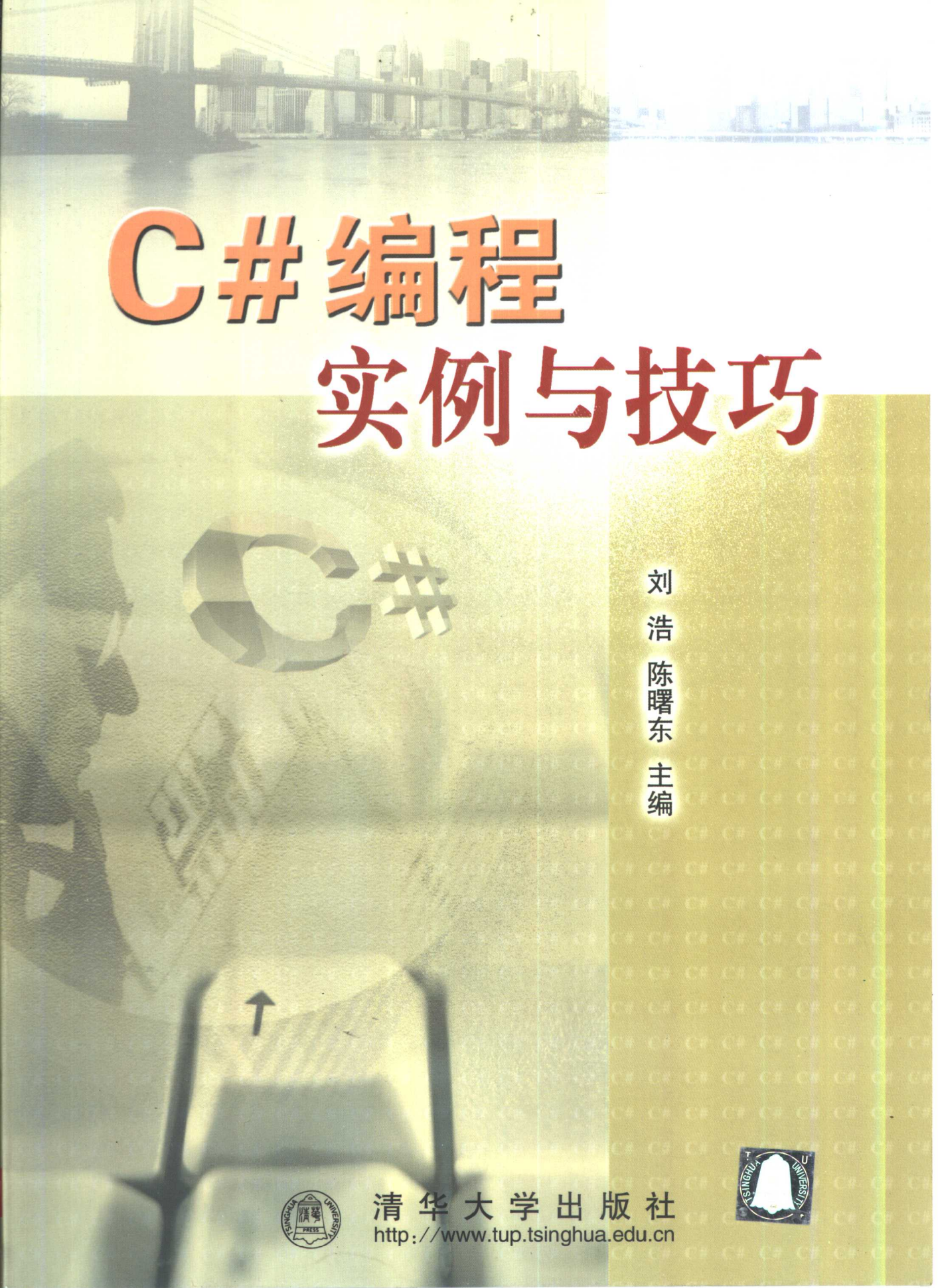

C#编程实例与技巧

刘浩 陈曙东 主编

清华大学出版社
http://www.tup.tsinghua.edu.cn

C#编程实例与技巧

刘　浩　陈曙东　主编

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书针对微软 Visual Studio. NET 的用户,详细介绍了 C# 在 WinForm、图形图像处理、多媒体、数据库、网络编程等方面的使用过程、方法和技巧。

全书通过大量的实例重点向读者介绍了 C# 的应用。

本书运用大量新颖、实用的实例,让读者可以在不断地实践操作中学习、掌握 C# 的具体应用方法和编程技巧。

本书既可以作为广大 C# 初学者入门的自学读物,也可以作为学习 C# 的程序员的实例练习集。

书　　名:C# 编程实例与技巧
作　　者:刘浩　陈曙东　主编
出 版 者:清华大学出版社(北京清华大学学研大厦 A 座,邮编 100084)
责任编辑:宋　韬
印 刷 者:北京市清华园胶印厂
发 行 者:新华书店总店北京科技发行所
开　　本:787×1092　1/16　印张:19.75　字数:456 千字
版　　次:2002 年 2 月第 1 版　2002 年 2 月第 1 次印刷
书　　号:ISBN 7-302-05173-9/TP·3033
印　　数:0001-5000 册
定　　价:30.00 元

前　言

微软最新推出的面向对象编程语言 C#（读作“C sharp”）是一种基于 XML 语言，但又增强了 Web 服务能力的运行于.NET 平台的新型语言。.NET 平台提供的工具和服务能充分发掘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。因此，C# 的编译器是.NET 平台上最经得起检验、优化程度最好的编译器。

C# 吸取了 C、C++ 的精华。熟悉 C、C++ 语言的编程人员可以很快精通 C#，C、C++ 开发者能使用现有的技术，快速地构建基于 XML 的系统框架。

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、底层代码的控制，实现跨语言、跨平台的因特网远程调用。因此，开发者可以构建复杂的商务系统。C# 将 VB 的所有优点引入到 C++ 中，使程序员能用更少的代码完成更强大的功能，同时又减少错误的出现率。此外，C# 基于大多数程序员和集成商都已具有 C、C++ 的基础知识，可以缩短开发周期、节约开发费用，使产品跟上 Web 领域快速发展的步伐。所以，C# 无疑是构建从高层次的商务对象到系统级应用范围的最佳选择。

作者也是从 C# 的入门者成长起来的，深感学习 C# 时，如果有一本好书是最大的帮助，但却来之不易。首先，C# 是一门新兴的语言，参考资料不多；其次，普通教程着眼于 C# 的理论，常常读过了之后，不知道如何编写出好的程序。在这种情况下，《C# 编程实例与技巧》面世了，内容涵盖了 C# 在图形图像处理、多媒体、数据库、网络编程等方面的应用。通过具体的实例和详细的讲解给读者一种耳目一新的感觉。书里有作者的挫折和成功、经验和教训，相信读者可以从中受益。

本书突破了以往图书的传统写法，在讲述 C# 的每一项功能时，都让读者按照书中的步骤自己做一个小例子，待操作完之后，再详细讲述其中的重点和难点，让读者在具体的操作过程中充分体会功能的特点和用法，将 C# 的理论和实践相结合，避免实际应用中的眼高手低。

通过本书的学习，读者可以在短时间内学会 C# 编程，熟悉其应用的各个方面，达到中级以上的水平；对于已经有一定功底的读者，本书同样是一本较好的参考资料。

本书的宗旨在于：让读者成为一个精通 C# 的高手。对于从未接触过 C# 的读者，可以从内容详实的例子入手，轻松掌握 C# 的功能。对于接触过 C# 的读者，可以从本书的新功能介绍、技巧讲述和使用经验中受益。

在编写本书的过程中，得到了来自各方面的帮助和支持。在此衷心感谢以下参编人员：彭勇、胡正万、巫宗芳、张炯明、陈梅、张家彬、胡涛、孙忠、刘小伟、邓勇、欧阳劲、张云勇、卢军、唐寅、邹思轶、张炯明、李彬、卫星、贺玉龙、陈明、李宋琛、邓海、吴文锦、巫文斌、刘青松、田茂敏等。在本书的写作过程中，他们给了我很多技术和友情上的帮助。

如果读者愿意参加“C# 编程实例与技巧”的培训学习，或在学习过程中发现问题和有更好的建议，欢迎来函来电联系。通讯地址：成都四川大学（西区）建筑学院成都博嘉科技资讯有限公司，邮编：610065。电话：(028)5404228；E－mail：bojiakeji@163.net。

作　者

目　录

第1章　C#概述

本章要点：

本章首先介绍了C#的推出背景和C#语言的特点，在读者对C#有一个大体印象之后，再详细介绍C#的运行环境——微软.NET平台的基本框架。通过本章的学习，读者可以了解微软.NET平台的工作机制及其有关概念。

本章主要内容：

- ☐ 微软为什么推出C#
- ☐ Microsoft.NET框架

1.1 微软为什么推出C#

在过去的20年中,C和C++已经成为开发商务软件和企业软件使用最为广泛的编程语言。这两种语言为开发者提供了极大的灵活性,但这种灵活性是以延长开发周期为代价的。例如与Visual Basic比较,C和C++应用程序相对来说需要较长的开发时间。由于开发复杂程度高,开发周期长,许多C和C++程序员努力在寻找一种能在功能和开发周期上提供更好的编程语言。

有几种编程语言是通过牺牲C和C++的灵活性来缩短开发周期的,例如,省略低级代码控制。这样的解决方案对开发者的约束太多并且通用性很差,与已有的系统很难相互操作,并且与当前的Web设计方法不能很好吻合。

对C和C++程序员来说,理想的解决方法是寻找一种既可以进行快速开发,又可以访问所有潜在平台的编程语言。并且,这种编程语言的开发环境能完全与新的Web标准同步,容易与现存的应用系统集成,同时,C和C++程序员还希望必要时可以使用这种语言来编写底层代码。

微软为了解决上述问题,提出了一种名为C#的语言(读作:C sharp)。C#是从C语言和C++语言派生的一种简单的、现代的、面向对象的、类型安全的程序设计语言,它使开发人员能够在微软新的.NET平台上快速建立广泛的应用。它提供的工具和服务能充分发掘系统的计算能力和通信能力。在构建从高级商务对象到系统级应用的各种不同组件时,由于C#具有优良的面向对象设计特点,已成为首要的选择。使用简易的C#语言,所有的组件都可以被转换为Web服务,从而实现跨语言、跨平台的因特网远程调用。

C#的设计还使得C++程序员进行快速开发成为可能,而不用牺牲C++已有的功能和控制能力。通过这种继承,C#保持了与C和C++的高度一致。开发者只要熟悉C和C++语言就可以快速地掌握C#,并写出更多的C#应用程序。

从经济上考虑新的Web要求所有开发者在更短的时间内开发出更多的程序版本,使产品能跟上Web领域快速发展的步伐。微软在进行C#的设计时考虑到了这个因素,它的设计使开发者可以使用更少的代码完成更强大的功能,同时减少错误的出现率。

下面是C#的各种特点:

- 遵循新的Web设计标准。
- 消除重要的编程错误。
- 依赖内建的转换支持降低开发成本。
- 广泛的协同工作能力。

1.1.1 遵循新的Web设计标准

新的应用程序开发模型意味着越来越多的解决方案需要使用新的Web标准,如HTML、XML和SOAP(simple object access protocol,简单对象访问协议)。现存的开发工具都是在Internet之前或之初开发的,因此,它们总是不能很好地适应新的Web技术。

C#程序员能用一个扩展的框架构建微软.NET平台上的应用。C#提供内建的支持服务,可以将组件转换为能在Internet上运行,并且可以被任何平台上的任何应用调用的组件。对程序员来说,这个Web服务框架更突出的优点在于它能够将现存的Web服务看起来像在对本地的C#对象进行操作。因此,为开发者在利用已有的面向对象的编程技能以及现存的Web服务方面提供了更大的选择余地。

C#具有的其他更优秀的特点使其成为主要的Internet开发工具。例如,XML是新出现的在Internet中传递结构化数据的方法,但是,这种数据集通常很小。为了提高性能,C#允许XML数据被直接映射到一个结构数据类型而不是一个类。在数据量不大时,这是一种十分有效的处理方式。

1.1.2 消除重要的编程错误

即便是系统级的C++程序员都避免不了最简单的错误,例如,忘记初始化一个变量。这些简单的错误常常导致不可预见的问题,它们长时间隐蔽,不易发现。但是,一旦程序投入生产运行,要排除哪怕是最简单的编程错误,都要付出非常昂贵的代价。

C#在设计时排除了大多数普通的C++编程错误,例如,垃圾清理减轻了程序员自己管理内存的负担;在C#中变量是自动被环境初始化的;变量类型也是安全的。

C#实施了最严格的类型安全来保护自身和垃圾收集器,因此,在C#中必须遵守关于变量的一些规定:

- 不能使用未初始化的变量。对于对象的成员变量,编辑器负责将它们清零,局部变量由程序员自己负责。如果使用了未经初始化的变量,编译器会提醒程序员。这样做的好处是:可以摆脱因使用未初始化变量而得到一个错误的结果。
- C#不支持不安全的类型指向。不能将整数指向引用类型(如对象),当进行下行指向时,C#会验证指向的有效性。也就是说,导出对象确实是从要将它下行指向的类型中导出的。
- 边界检查是C#的一部分。当数组实际上只有n-1个元素时,不可能使用它"额外"的数组元素n,因为这样就会将未经分配的内存重写。
- 算术运算可能溢出结果数据类型的范围。C#允许在应用级或者语句级对这类操作进行溢出检查,当溢出发生时会产生一个异常。
- C#中传递的引用参数使类型可以使用以上安全机制,最终使程序员使用C#语言更加容易地进行用于解决复杂业务问题的程序的开发和维护。

1.1.3 依赖内建的转换支持降低开发成本

更新软件组件容易出错,修订代码版本无意中可能会改动程序的语义,为了帮助程序员解决这个问题,C#语言中包含了转换支持。例如,与C++和Java不同,C#中的方法重载必须显式地进行,这有助于防止代码错误,同时保留了转换的灵活性。再比如,C#支持本地接口和接口继承,使复杂的框架得到了发展。

这些特点使一个工程的后继版本的开发方法更加健全,同时降低了后继版本的整体

开发成本。

1.1.4 广泛的协同工作能力

随着公司制作商业计划的水平越来越高,抽象的商业过程与实际软件实现之间的紧密连接已成一种必然趋势。但大多数语言工具没有一种简单易行的方式来进行业务逻辑和代码之间的连接,例如,开发者可能使用代码注释说明哪个类构成主要的抽象业务对象。

C#语言允许分类和扩展,这种分类和扩展可以应用于任何对象的元数据。一个工程的构建者首先定义特定领域的属性,并将它们应用到任何语言的类元素和接口中。然后开发者编程测试每个元素的属性,这样,就简化了自动化工具的编写。而自动化工具会保证每个类和接口被正确地表示为特定的抽象商业对象,从而简化了基于对象的特定属性的创建过程。定制元数据和程序代码之间的紧密结合,有助于实现商业过程以及实现它们之间最佳的映射。

可管理的类型安全性环境适用于大多数企业应用。但是,根据经验可知,由于性能问题以及应用程序的接口问题,有些应用仍然需要"本地"代码。在这样的情况下,开发者使用C++来开发无疑是最佳的选择。

C#通过下面的方式解决了这个问题:

- 支持组件对象模型(COM)的本地化,支持基于Windows APIs的调用

在C#中,每一个对象自动地变成一个COM对象。开发者不需要再显式地实现其他COM和IUnknown接口,因为C#将它们内置了。同样,C#编程能本地化地使用现存的COM对象,而不关心它们是用什么语言创建的。

对于需要这种功能的开发者来说,C#另外一个很好的特点是可以任意地调用本地API。C#实现COM支持和本地API访问支持,这两个功能的目的就是在不脱离C#编译环境的前提下,为开发者提供更强大的功能和控制能力。

- 允许有限制地使用本地指针

C#具有传统的C/C++特点。在一个带有特定标记的代码块中,C#允许开发者使用指针管理内存和指针算法。这是其他开发环境所没有的特点,它意味着C#程序员能使用现存的大量C和C++代码,而不用丢弃它们。

1.2 Microsoft.NET框架

C#是微软.NET平台的首推语言,所有的特点都是根据.NET平台的要求来设定的。为了更深入地了解C#的内涵及其运行的基础,有必要将微软.NET的框架结构做一个整体的了解。另外,本书也是根据.NET框架的结构进行编排和组织的。因此,这一节就对.NET平台的框架结构做一个简要的介绍。

1.2.1　微软.NET Framework 体系

当今 Internet 的发展要求将应用程序进行集成，将运行在不同操作系统上的、使用不同编程语言的对象模板建立的不同应用程序，转化为易于使用的网络应用程序。建立在 HTTP 和 XML 开放的网络标准上的网络服务是最合适的。网络服务是基于网络的分布式应用程序的基本构造模块。这些程序是以平台、对象模板和多语言方式构建的，也是微软可编程网络理念的基础。但是，只支持标准协议是不够的，还必须有途径来生成、部署、扩展和维护这些网络服务，这正是 Microsoft.NET 框架要解决的问题。

构建 Microsoft.NET 框架的目的就是使建立网络应用程序和网络服务更容易。图 1－1显示了 Microsoft.NET 框架的体系。建立在操作系统最上层的服务，是管理运行时代码需求的公用语言运行环境(common language runtime，CLR)，这些代码可以用任何现代编程语言编写。运行环境提供了许多服务，这些服务有助于简化代码和应用程序的开发，同时也将提高应用程序的可靠性。.NET 框架体系中包括一套可被开发者用于任何编程语言的类库。在此之上是许多应用程序模板，这些模板为开发网络站点和网络服务提供特定的高级组件和服务。这些层次都将在后续章节中描述。

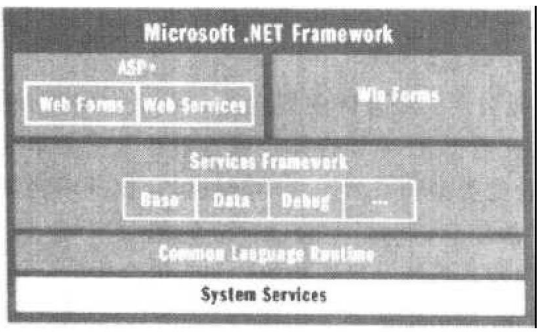

图 1－1　微软.NET Framework 体系

1.2.2　公用语言运行环境

.NET 框架体系中，建立在操作系统最上层的服务是管理运行时代码需求的公用语言运行环境。

运行环境(runtime)调入并运行各种使用编程语言所写的代码。以运行为目标的代码被称为受控(managed)代码，受控代码意味着只在内部执行的代码与运行自身间存在已定义好的合作契约。将生成对象、调用方法等任务委托给运行环境，这使得运行环境能为可执行代码增加额外的服务。

运行环境具有交叉语言集成、自描述组件、简单配制、版本化和集成安全服务等特点。下面将对每一个特点进行简要的介绍。

运行环境使用一种能表达大部分现代编程语言语义的新的通用类型系统。通用类型系统定义了一套标准类型及生成新标准的规则。运行环境知道怎样生成、执行这些类型。

编译器和解释器在进行定义类型、管理对象以及进行方法调用时使用运行环境服务，而不是使用工具或特定于语言的方法。

类型系统的主要设计目标是使多种语言能深度集成。一种语言所写的代码能够继承另一种语言所写的类的实现，一种语言所写的代码抛出的异常能被另一种语言所写的代码捕获，同时，调试、剖析之类的操作完全封闭，无需考虑代码所用的语言。这就意味着编写可重用类库的开发者，不需要再为每一种编程语言或编译器生成一个版本，并且使用类库的开发者也不再受到开发所使用类库时用何种编程语言的限制。

自描述组件简化了开发和配制，并提高了系统的可靠性。许多由运行环境提供的服务由元数据和用于补充可执行代码的信息所驱动。因为所有的信息都储存在一起，只有可执行的代码才被称为自描述组件。

自描述组件的一个主要优点是，使用它们并不需要其他文件。类的定义不需要单独的头文件，可以从组件自身获得元数据对类的定义。跨语言或过程边界访问组件并不需要各自的 IDL 文件或类型文件，所必需的信息已存在于元数据之中。要识别开发者表示的服务属性，并不需要展开各自的配制信息。最主要是由于元数据是在编译过程中由源代码生成，并与可执行代码储存在一起，它将永远和可执行部分同步。

除了改善对单个组件的配制外，Microsoft.NET 框架定义了一个应用程序配制模板，以解决定制应用程序安装和 DLL 版本化（通常被称为“DLL Hell”）这一复杂过程的问题，运行环境提供了支持这个模板的服务。

Microsoft.NET 框架引入了组合体的概念。一个组合体是一组资源和类型，并包括有关这些资源和类型的元数据，一个组合体也就是被作为一个单元配制的。元数据被称为组合体的清单，它包含像类型和资源表之类能被组合体外看得见的信息。这个清单也包括有关从属关系之类的信息，例如，组合体建立时的版本号。开发人员可指定版本策略，以指示运行环境是否装入系统上已安装的依赖于组合体的最新版本，或装入一指定版本或在编译时使用的版本。

某软件组件的多个拷贝可以存在于同样的操作系统上。然而，通常只有其中的一个拷贝能被操作系统注册、调入内存并执行。对系统来说，定位和调入内存的策略是全局性的。.NET 框架公用语言运行环境增加了所必须的体系架构，以支持管理组件的定位、调入的每个应用程序策略，这通常称为“并行配制”。

组合体可以被一个应用程序独有，也可以被多个应用程序共享。一个组合体的多个版本可以同时配制在同一台机器上。应用程序配制信息定义了到何处去查找组合体，这样，运行环境就能为同时运行的两个不同的应用程序装入同一组合体的不同版本。这就消除了由组件版本的不兼容性引起的问题，提高了系统整体的稳定性。如果必要，管理员可以为配制的组合体增加配制信息，例如，一个不同的版本策略，但编译时提供的原始信息永远不会丢失。

因为组合体是自描述的，所以，并不需要在系统上进行显式注册。应用程序的配制甚至可以简单到只需将文件拷贝到目录中即可。如果为了使应用程序能够运行，必须安装未经组织过的组件，会稍微复杂一点。配制信息保存在可被任何文本编辑器编辑的 XML 文件中。

最后，运行环境还提供完整的安全服务，以确保未经授权的用户不能访问机器上的资

源，并且代码不会执行未经授权的动作，从而提高了系统整体的安全性、可靠性。由于运行环境用于装入代码、生成对象和执行方法调用，所以，当受控代码装入内存执行时，运行环境能进行安全检查，强化安全策略。

Microsoft.NET 框架不仅规定代码访问安全，还规定基于角色的安全。通过代码访问安全机制，开发人员能为应用程序指定完成工作所必需的权限，例如，代码或许需要写文件或访问环境变量的权力。这类信息和有关代码标志的信息是一起存储在配制级上的。当代码装入内存及执行方法调用时，运行环境验证是否能给予代码所要求的权限。如果不能，系统将记录一条安全冲突信息。给予权限的策略被称为信任策略，信任策略由系统管理员建立，是建立在关于代码的证据基础上的。所谓代码的证据指的是，代码是谁发布的，从什么地方获得的，以及在组合体中找到的代码标志和它要求的权限。开发人员可以指定他们不需要的权限，以防止其他人恶意使用他们的代码。如果所需要的权限依赖于直到运行时才会获取的信息，那么就可写入安全检查。

除了代码访问安全外，运行环境还支持基于角色的安全。基于角色的安全建立与代码访问安全一样的权限模板，只是这些权限是建立在用户的身份上的，而不是建立在代码的标志上。角色表明了用户所属的类，并且可以在开发和配制阶段定义。给予权限的策略被分配到每个预定义的角色。在运行时，用户的身份被确定，代码将代表这个身份运行。运行环境决定用户是哪个角色的成员，然后给予基于这个角色的权限。

1.2.3　服务框架的主要类库

如图 1－1 所示，在公用语言运行环境之上是服务框架，此框架提供能被任何现代编程语言调用的类。所有的类都遵循一套命名和设计方针，可以大大减少开发人员学习上的弯路。图 1－2 显示了服务框架中的一些主要类库。

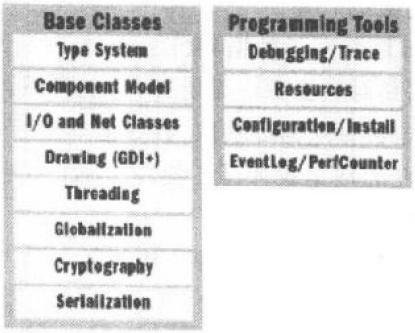

图 1－2　服务框架中的类库

服务框架包括一套开发人员希望在标准语言库中存在的基类库，例如，集合、输入/输出、字符串及数据类。另外，基类库提供访问操作系统服务，如图画、网络、线程、全球化和加密的类。服务框架也包括数据访问类库和开发工具，如调试和剖析服务使用的类。

这一节并没有详细讨论所有的类，重点只讲解数据访问类，因为大多数网络服务需要对数据的访问。如果需要关于服务框架类库的附加信息，可以参考 Microsoft.NET Framework SDK。

几乎所有的网络服务都需要查询和更新永久性数据,不论是以简单文件,还是以相关数据库,或是以其他存储类型存在。为了提供对数据的访问,服务框架包括 ActiveX 和 Data Objects+(ADO+)类库。如同其名字所暗示的那样,ADO+是由 ADO 发展而来的。ADO+被设计为基于网络的可扩展的应用程序和服务提供数据访问服务。ADO+提供指针型的数据访问,同时也为更适合于把数据返回到客户端应用程序的无连接的数据模板提供高性能的 APIs 流。ADO+的体系结构如图 1-3 所示。它表明任何数据,不论这些数据是如何存储的,它们都以 XML 或相关数据的格式进行操作。不论是哪一种操作格式,在一个给定的时间点上对应用程序都是最合适的。

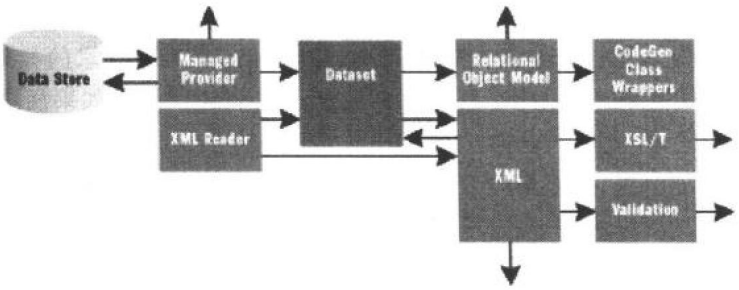

图 1-3 ADO+体系结构

ADO+定义了那些链接数据仓库、对数据仓库发送命令和从中获取结果的类。这些类由管理数据提供程序(管理数据支持程序)实现。ADO+中的链接和命令对象,看上去与 ADO 中是一样的,并且一个名为 DataReader 的新类,提供了通过高性能 API 流获取结果的能力。虽然 DataReader 在功能上与前向的、只读的 ADO 记录集(recordset)是等同的,但是,DataReader 被设计用来最小化内存中生成的对象数量,以提高性能,避免垃圾积累。在.NET Framework 中包含了针对 Microsoft SQL Server 的管理数据提供程序,以及可通过 OLE DB 访问的任何数据仓库。

ADO+的一个主要创新是引入了数据集(Dataset)。一个数据集是内存中提供数据关系图的高速缓冲区。数据集对数据源一无所知,它们可以由程序或通过从数据仓库中调入数据而生成、填充。不论数据从何处获取,数据集都是通过使用同样的程序模板而被操作的,并且使用相同的潜在的数据缓冲区。使用.NET 平台的开发人员能够用数据集代替传统 ADO 中无链接的记录集。

管理数据提供程序为数据仓库和数据集公开一名为 DataSetCommand 的接口对象。DataSetCommand 使用 ADO+链接和命令从数据仓库中填充数据集,并把数据集中发生的变化解析到数据仓库中。

就如 DataReaders 显示了对相关数据的有效的流访问一样,XMLReaders 显示了对 XML 数据的流访问。开发人员使用 DataNavigator,可以滚动和编辑内存中的 XML 文档。DataNavigator 在功能上与 W3C Document Object Model(DOM)是一样的,但它更有效,并提供了能很好映射关系数据表的对象模板。DataNavigator 支持 Xpath 语法,以对数据流进行导航。ADO+为那些希望继续使用 DOM 作为 XML 对象模板,而不是使用更有效的 DataNavigator 模板的开发人员提供了一个 XMLDocument 类。

由于所有数据都可被看作 XML,所以,开发人员可以对任何数据使用转换和确认服

务。ADO + 定义了一个 DataNavigator，生成一个新的 XMLReader 的通用转换体系。.NET 框架提供了一个支持 W3C XSL Transformations(XSLT)细则的特殊转换组件。同时，ADO + 提供了一使用 XML 简图确认 XMLReader 的确认引擎。ADO + 还支持通过 DTDs、XSD 或 XDR 定义的简图。

1.2.4　表单应用模板

从概念上讲，在服务框架的最上面是两个应用程序模板：Windows 应用程序模板和网络应用程序模板。尽管微软.NET 框架主要是用在开发网络服务和网络应用程序的一种途径上，但框架也可用于开发较传统的基于 Windows 的应用程序。当然，这些应用程序也能使用网络服务。

编写 Windows 客户应用程序的开发人员，可使用 Win 表单应用程序模板以利用 Windows 丰富的用户接口特点，包括现在的 ActiveX 控件和 Windows 2000 的新特点，如透明的、分层的和浮动窗口。可以选择传统的 Windows 或网络外观，它与现在基于 Windows 的表单非常相似。开发人员会发现 Win 表单的可编程模板和对设计阶段的支持非常直观。

Win 表单利用 Microsoft.NET 框架运行环境，以减少基于 Windows 的客户应用程序的开销。只要应用程序和组件是用 Win 所写或被 Win 表单应用程序使用，它们就能被框架安全模板在客户机上安全地执行。如果以这种方式使用或执行，那么某人从 Internet 下载的游戏就不会破坏配制信息和数据，否则会自动地给用户地址薄里的每一个人发送电子邮件。

Microsoft.NET 框架装配模板简化了应用程序的配制和版本化。应用程序可被配制为使用它们在编译和测试所用的共享组件，而不是使用恰好在客户机器上安装的任何版本的组件。这提高了应用程序的可靠性，减少了应用程序所支持调用的主要因素：用户接口控件和其他共享组件版本的不兼容性。

1.2.5　ASP + 网络应用模型

ASP + 网络应用模型是建立在 Microsoft.NET 框架上网络应用程序共享的一个通用应用程序模板。在这个模型中，网络应用程序是一套起源于基本 URL 的 URLs。因此，它包含用于生成在浏览器中可观看网页的网络应用程序和网络服务。在本小节中，将详细介绍名为 Active Server Pages + (ASP +)的网络应用程序可编程模板，如图 1 - 4 所示。

从名字可以看出，ASP + 由活动服务器页面发展而来。ASP + 利用公用语言运行环境和服务框架网络应用程序提供了一个可靠的、自动化的和可扩展的主机环境。ASP + 也受益于公用语言运行环境集成模板，简化了应用程序的配制。另外，它提供简化应用程序开发的服务(如状态管理服务)以及高水平的编程模板(如 ASP + Web Forms 和 ASP + Web Services)。

ASP + 的核心是 HTTP 运行环境——一个高性能的用于处理基于低级结构的 HTTP 请求的运行环境。而它基于的结构与 Microsoft Internet Information Services(IIS)所提供