普通高等应用型院校“十二五”规划教材

软件工程专业

.NET框架程序设计

主　编　胡晓宏　薛京丽　副主编　李　卓　张玲玲　赵险峰

中国水利水电出版社
www.waterpub.com.cn

普通高等应用型院校“十二五”规划教材

.NET 框架程序设计

主　编　胡晓宏　薛京丽

副主编　李　卓　张玲玲　赵险峰

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书是一本非常实用的学习.NET 框架程序设计的教材，主要介绍基于 C#的.NET 框架技术，带领读者去探索、领悟一个关于.NET 平台核心技术的思想体系；介绍基于.NET 框架的应用程序的开发，解释如何开发面向.NET 框架的应用程序，包括.NET 框架下应用程序的执行和编译原理以及.NET 框架类库中的核心类型和使用方法。本书遵循循序渐进的教学原则，注重能力的培养，结合实际讲解理论，为配合教学和学习，本书为每个知识点都配了必要的实例，力求通过实例让读者掌握 C#的.NET 框架程序设计技术。本书源于丰富的教学实践和项目开发实践，适合于边讲边练、做中学的课堂教学。

本书可以作为计算机专业的教材及教学参考书，也可以作为计算机开发应用人员的参考书。

图书在版编目（CIP）数据

.NET框架程序设计 / 胡晓宏，薛京丽主编. -- 北京：中国水利水电出版社，2015.6
普通高等应用型院校“十二五”规划教材
ISBN 978-7-5170-3268-7

Ⅰ. ①N… Ⅱ. ①胡… ②薛… Ⅲ. ①计算机网络－程序设计－高等学校－教材 Ⅳ. ①TP393.09

中国版本图书馆CIP数据核字(2015)第131956号

策划编辑：石永峰　　责任编辑：张玉玲　　封面设计：李　佳

书　名	普通高等应用型院校“十二五”规划教材 .NET 框架程序设计
作　者	主　编　胡晓宏　薛京丽 副主编　李　卓　张玲玲　赵险峰
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（发行部）、82562819（万水）
经　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	铭浩彩色印装有限公司
规　格	184mm×260mm　16 开本　14.75 印张　364 千字
版　次	2015 年 6 月第 1 版　2015 年 6 月第 1 次印刷
印　数	0001—3000 册
定　价	30.00 元

前　言

随着网络计算时代的到来，各种应用于网络服务的计算机语言、操作系统和开发工具应运而生。C#是在 C、C++、Java 语言基础之上开发的运行于.NET 平台为适应 Internet 和各类网络应用而设计的编程语言，它综合了 C、C++、Java 以及其他高级语言的优点，是一种类型安全、完全面向对象的编程语言。随着.NET 技术的普及，C#必将成为开发 Internet 和企业应用程序的首选程序设计语言。

在.NET 平台中，.NET 框架占据着核心的位置，它是整个.NET 平台的关键支持。学习.NET 框架是学习 C#程序设计语言的一个提高，.NET 技术浩如烟海，从微观入手、从底层入手是掌握软件技术的重要方法，学习.NET 底层框架技术可以从整体上把握.NET 软件开发的方法，如果没有对.NET 框架的深刻把握，学习再多的.NET 应用程序模型开发技巧都将是徒劳。因此不管是学习 Windows 窗体、ASP.NET Web 窗体还是学习 XML Web 服务，笔者都建议大家应先从.NET 框架开始迈出坚实的一步——探微而知著。

本书是一本非常实用的学习.NET 框架程序设计的教材，主要介绍基于 C#的.NET 框架技术，带领读者去探索、领悟一个关于.NET 平台核心技术的思想体系；介绍基于.NET 框架的应用程序的开发，解释如何开发面向.NET 框架的应用程序，包括.NET 框架下应用程序的执行和编译原理以及.NET 框架类库中的核心类型和使用方法。本书遵循循序渐进的教学原则，注重能力的培养，结合实际讲解理论，为配合教学和学习，本书为每个知识点都配了必要的实例，力求通过实例让读者掌握 C#的.NET 框架程序设计技术。本书源于丰富的教学实践和项目开发实践，适合于边讲边练、做中学的课堂教学。

本书由胡晓宏、薛京丽任主编，李卓、张玲玲、赵险峰任副主编，具体编写分工如下：第 1～3 章和第 5 章由胡晓宏编写，第 8 章和第 9 章由薛京丽编写，第 13 章由李卓编写，第 12 章由张玲玲编写，第 4 章和第 7 章由赵险峰编写，第 10 章由郑慧编写，第 11 章由尹健慧编写，第 6 章由刘红杰编写。

由于时间仓促和编者水平有限，书中难免有不妥之处，敬请广大读者批评指正，编者电子邮箱：bhhxh69@163.com，欢迎来信。

编　者

2015 年 5 月

目　录

第 1 章　.NET 体系结构

.NET 是 Microsoft 的一种开发模型，它使软件变得独立于平台和设备，并且支持利用 Internet 上的数据编程，.NET 可以使程序运行于各种异构平台。.NET 平台为创建新一代分布式 Web 应用程序提供了所有工具和技术（表示技术、构件技术和数据库技术）。.NET 平台支持标准的 Internet 协议，包括 HTTP（超文本传输协议）、XML（可扩展标记语言）和 SOAP（简单对象访问协议），从而实现了异构系统间应用程序的集成和通信。.NET 框架是.NET 的基本体系结构，它提供了具体的技术和服务（手段），在本章中将学习.NET 的体系结构。

本章主要内容：

- .NET Framework 的构成及其组件的概念。
- .NET Framework 的类库和命名空间之间的关系。

1.1　.NET 简介

.NET 是微软在本世纪初，应网络分布式运算需求而推出的一个应用程序开发和运行平台规范。该规范内容相当广泛，包含了诸如组件格式、编程语言、标准类和工具等各个方面。为推广和使用这个规范，作为该规范的首倡者，微软在公布这个规范的同时也推出了该规范在 Windows 平台上的一个实现——.NET Framework。.NET 框架是.NET 的基本体系结构，它提供了具体的技术和服务。现在人们所说的.NET，通常就是指微软的这个实现以及这个实现所包含的各项技术。

1.2　.NET Framework 概述

.NET Framework 是支持生成和运行下一代应用程序和 XML Web Services 的内部 Windows 组件。.NET Framework 旨在实现以下目标：

（1）提供一个一致的面向对象的编程环境，而无论对象代码是在本地存储和执行，还是在本地执行但在 Internet 上分布，或者是在远程执行的。

（2）提供一个将软件部署和版本控制冲突最小化的代码执行环境。

（3）提供一个可提高代码（包括由未知的或不完全受信任的第三方创建的代码）执行安全性的代码执行环境。

（4）提供一个可消除脚本环境或解释环境的性能问题的代码执行环境。

（5）使开发人员的经验在面对类型大不相同的应用程序（如基于 Windows 的应用程序和基于 Web 的应用程序）时保持一致。

（6）按照工业标准生成所有通信，以确保基于.NET Framework 的代码可与任何其他代码集成。

如果要使用 C#高效地开发应用程序，理解.NET Framework 就非常重要，.NET Framework 包括以下组件：

- 公共语言运行库（Common Language Runtime，CLR）。
- .NET Framework 类库（Framework Class Library，FCL）。
- 数据库访问组件（ADO.NET 和 XML）。
- 基于 ASP.NET 编程框架的网络服务（Web Services）和网络表单（WebForms）。
- Windows 桌面应用界面编程组件（WinForm）。

.NET Framework 的体系结构如图 1-1 所示。

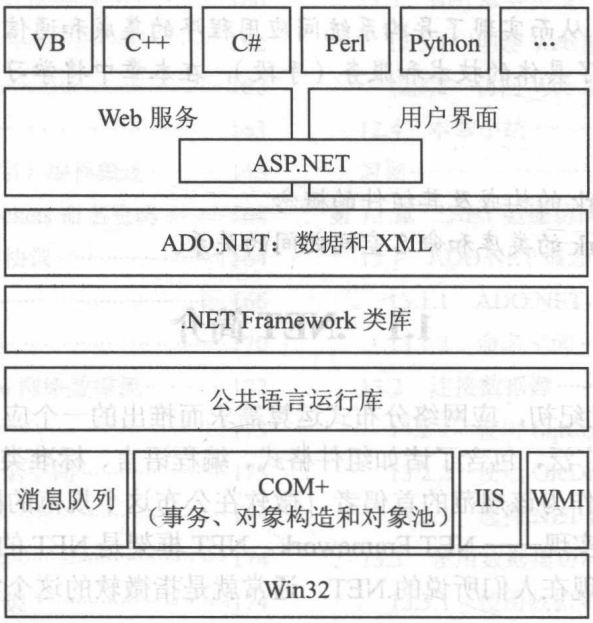

图 1-1　.NET Framework 的体系结构

公共语言运行库（CLR）简化了应用程序的开发，并为应用程序提供了一个强大和安全的执行环境。公共语言运行库是 .NET Framework 的基础。可以将公共语言运行库看做一个在执行时管理代码的代理，它提供内存管理、线程管理和远程处理等核心服务，并且还强制实施严格的类型安全以及可提高安全性和可靠性的其他形式的代码准确性。这类似于Java的虚拟机。事实上，代码管理的概念是公共语言运行库的基本原则。以公共语言运行库为目标的代码称为托管代码，不以公共语言运行库为目标的代码称为非托管代码。

.NET Framework 类库是一个综合性的面向对象的可重用类型的集合，它包含数量庞大的类，提供创建各种应用程序所需的广泛功能，简化了.NET 应用程序的开发，开发人员还可以通过创建自己的类库来扩展它们。

在基于.NET 的应用程序中，要访问数据库，使用 ADO.NET 组件是目前的最佳选择，它为非连接编程模型提供了改进的支持，也提供了丰富的 XML 支持。

ASP.NET 是一个建立在公共语言运行库上的编程框架，可以使用 ASP.NET 在服务器上构建功能强大的 Web 应用程序。ASP.NET Web 窗体提供了易用且功能强大的方法来构建动态的 Web 用户界面（UI）。

.NET Framework 提供了一系列工具和类来构建、测试和分发 XML Web Service。

.NET Framework 支持 3 种类型的用户界面：Web 窗体（通过 ASP.NET 工作）、Windows 窗体（运行在 Win32 客户端上）和控制台应用程序。

.NET 框架中最重要的元素是公共语言运行库（Common Language Runtime，CLR），公共语言运行库管理用.NET 语言编写的可执行代码像 Java 的虚拟机一样，它是.NET 体系结构的基础。公共语言运行库激活对象并在其上实施安全检查，在内存中部署、执行并且回收。

1.3　公共语言运行库

公共语言运行库（CLR）是整个.NET 框架的核心，它为.NET 应用程序提供了一个托管的代码执行环境。它实际上是驻留在内存里的一段代理代码，负责应用程序在整个执行期间的代码管理工作，比较典型的有：内存管理、线程管理、安全管理、远程管理、即时编译、代码强制安全类型检查等。这些都可称为.NET 框架的生命线。

公共语言运行库的组成如图 1-2 所示。

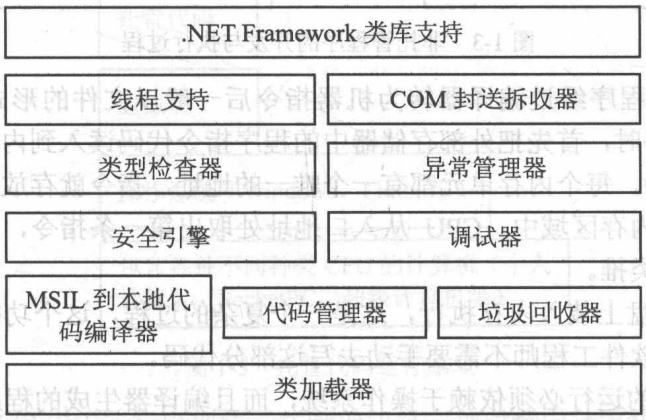

图 1-2　公共语言运行库的组成

下面对各个组成部分提供的功能进行简要介绍。

- 类加载器：管理元数据，加载和在内存中布局类。
- Microsoft 中间语言（MSIL）到本地代码编译器：通过即时编译把 Microsoft 中间语言转换成本地代码。
- 代码管理器：管理和执行代码。
- 垃圾回收器：为.NET Framework 下的所有对象提供自动生命期管理，支持多处理器，可扩展。
- 安全引擎：提供基于证据的安全，基于用户身份和代码来源。
- 调试器：使开发者能够调试应用程序和根据代码执行。
- 类型检查器：不允许不安全的类型转换和未初始化变量 MSIL 可被校验以保证类型安全。
- 异常管理器：提供和 Windows 结构化异常处理集成的异常处理机制。
- 线程支持：提供多线程编程支持。

- COM 封送拆收器：提供和 COM 组件之间的封送转换。
- .NET Framework 类库支持：通过和运行时集成代码来支持.NET Framework 类库。

实际上，CLR 代理了一部分传统操作系统的管理功能。通常将在 CLR 的控制下运行的代码称为托管代码，否则称为非托管代码。

1.3.1 非托管代码的运行原理

先来看一下 Windows 操作系统执行一个非托管程序的基本过程，如图 1-3 所示。

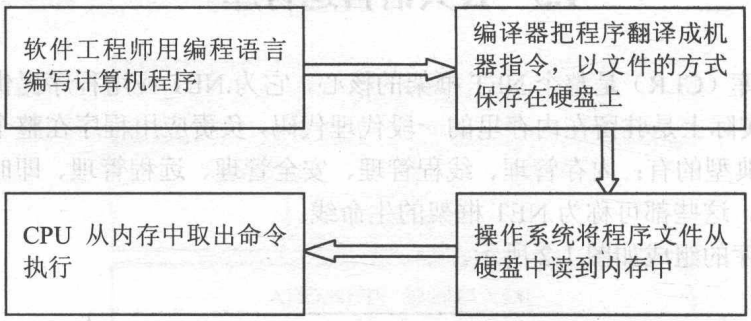

图 1-3 非托管程序的开发与执行过程

软件工程师写的程序经过编译器转为机器指令后一般以文件的形式保存在外部存储器中，当 CPU 执行程序时，首先把外部存储器中的程序指令代码读入到内存中。内存被分成许多块（称为内存单元），每个内存单元都有一个唯一的地址，指令就存放在以某个特定的地址（入口地址）开始的内存区域中。CPU 从入口地址处取出第一条指令，开始执行，然后再取出第二条指令，依此类推。

把一个程序从硬盘上装入内存执行，这是一个复杂的过程，这个功能由操作系统实现，开发具体应用程序的软件工程师不需要手动去写这部分代码。

可以看出，程序的运行必须依赖于操作系统，而且编译器生成的程序文件包含的是仅适用于特定 CPU 架构的机器指令。由于不同 CPU 架构的机器指令集不同，所以生成的这个可执行程序不能不加修改地在拥有不同 CPU 架构的计算机上运行。

以这种方式生成的机器指令代码就是前面提到过的非托管代码。某个包含非托管代码的程序如果不进行修改，不仅不能在不同 CPU 架构的计算机上执行，而且在不同的操作系统下也不能执行，比如一个 Windows 应用程序就无法直接在 Linux 下运行，反之亦然。图 1-4 很好地说明了非托管代码的运行原理。

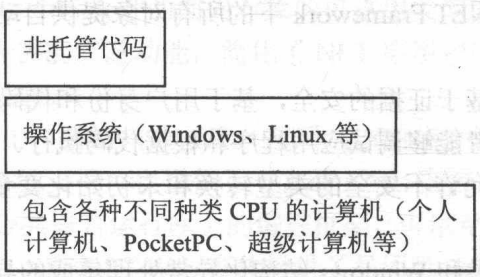

图 1-4 非托管代码的运行原理

1.3.2 托管代码的运行原理

显然，如果需要在拥有不同 CPU 架构的计算机和多种多样的操作系统上实现某一功能，必须针对每种操作系统和 CPU 架构编写特定的代码，这明显是一种重复和低效的劳动。那么程序能不能只写一次，却可以处处运行呢？

完全可以，这就是“跨平台”的设计思想。.NET 采用了这种设计思想，要支持跨平台这一特性，软件工程师写的程序经过编译器生成的结果就不能依赖于特定操作系统和特定 CPU 架构的机器指令，而必须是一种中间的、在各种操作系统和计算机硬件平台上都能执行的代码，这种代码.NET 称之为 MSIL（微软中间语言）指令。但程序最终还是要靠 CPU 执行，所以.NET 的 MSIL 指令仍然需要最终被翻译成本地 CPU 能执行的机器指令，这部分功能由一个运行在特定操作系统之上的软件系统来完成，这个软件系统被称为虚拟机（Virtual Machine，VM），CLR 就是.NET 虚拟机。这种运行在虚拟机之上的代码称为“托管代码”，MSIL 就是一种托管代码。

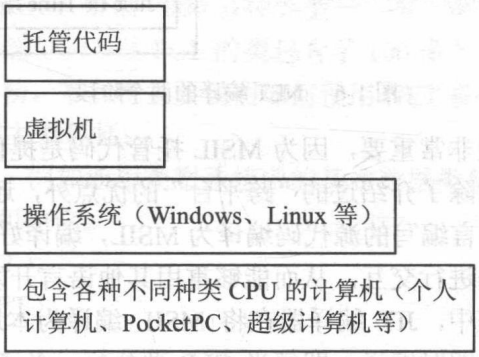

图 1-5 托管代码运行原理

图 1-5 所示为托管代码运行原理图，托管代码是一种对特定计算机和操作系统的中间语言，必须通过虚拟机才能将其转换为本地 CPU 能执行的机器指令。

如何跨语言应用呢？为了实现跨语言应用，微软采取了两项措施：一是开发了一种叫做 MSIL 的中间语言，它相当于是一种为虚拟机配套的汇编语言，该语言也叫做 IL（Intermediate Language）；二是提出了一个编译器的制作规范，这个规范叫做 CLS（公共语言规范，Common Language Specification）。

于是，那些各种不同的编程语言，只要配有按照 CLS 规范来制作的编译器，那么这种语言程序就能被编译成 IL 语言程序，也就能跨语言应用。

其实，只对语言进行统一还不能完全解决跨语言问题。因为，不同程序设计语言的差异不仅表现在语句的表达上，更重要是数据的内存布局。即不同语言的同一类型数据占用内存的大小和布局均有区别。例如，不同语言的函数在参数的传递顺序上就有所不同，所以以前在一个 Pascal 程序中调用 C 语言方法时就需要格外的小心，同理，在 C 程序中调用 Pascal 方法时也是如此。再例如，C 语言中的字符串是一个以 ASCII 码字符 NULL（在程序中通常用‘\0’表示）为结尾的字符数组；而在 Pascal 语言中，数组中的第一个字节就包含了字符串的长度，从而也就没有必要有一个类似的‘\0’结束标志。

综上所述，.NET 在使用 MSIL 进行了语言的统一之后，还需要对数据类型进行统一。为此，.NET 中就提供了一套统一的数据类型，这套数据类型就叫做公共类型系统（Common Type System，CTS）。也就是说，不同语言应用程序中的所有不符合 CTS 的数据类型，在经过编译器编译后都必须转换成统一后的类型，这样才能实现真正的跨语言应用。

由托管代码的运行原理可见，在.NET 中，编译被分为以下两个阶段（如图 1-6 所示）：

（1）把源代码编译为 MSIL。

（2）CLR 把 MSIL 编译为平台专用的代码。

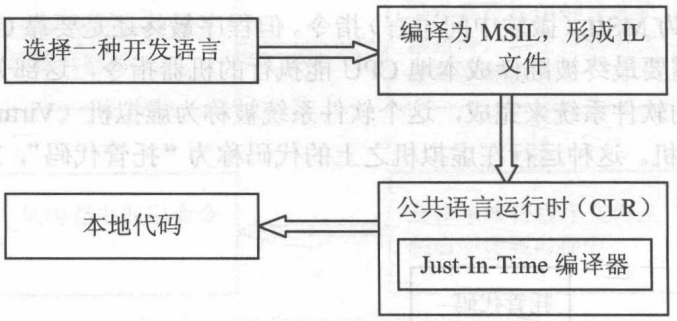

图 1-6　.NET 编译的两个阶段

这两个阶段的编译过程非常重要，因为 MSIL 托管代码是提供.NET 的许多优点的关键。托管代码与非托管代码相比除了介绍过的“跨平台”的优点外，还具有“跨语言”的优点。简言之，就是能将任何一种语言编写的源代码编译为 MSIL，编译好的中间语言代码可以与从其他语言编译过来的中间代码进行交互，从而能够重用其他语言开发的组件。

在第二阶段的编译过程中，JIT 编译器在将 MSIL 编译为本地代码时，并不是把整个 IL 文件一次编译完，而是采用即时编译，即每当 CLR 要执行一条函数调用指令时，由 JIT 编译器负责将这一函数所对应的 IL 指令动态编译成本地 CPU 可执行迟重的代码，CLR 同时将这些本地代码缓存起来，这样下次再调用此方法时就不需要重新编译了。这个过程要比一开始就编译整个应用程序代码的效率高得多，从而使得托管 IL 代码的执行速度几乎和内部代码的执行速度一样快。另外，JIT 编译器的编译过程是在运行时进行的，JIT 编译器会确切地知道程序运行在什么样的处理器上，因此可以针对特定的处理器来优化最后的可执行代码，而传统编译器的优化过程是独立于代码所运行的特定处理器的，例如 VS 6.0 为一般的 Pentium 机器进行了优化，所以它生成的代码就不能利用 Pentium III 处理器的硬件特性，传统编译器生成的代码就不能充分利用特定处理器的硬件特性，托管代码与非托管代码相比在性能上也有所提升。

另外，与非托管代码相比，托管代码还具有垃圾回收功能和代码访问安全性等优点，这部分内容会在后面章节作详细的介绍，所以这里只简单地介绍：垃圾回收功能就是为托管代码分配的内存，不需要程序员通过编程来释放它（例如 C++程序员利用 new 运算符分配内存时，一定要记得在不需要使用内存时用 delete 运算符将其释放掉），CLR 会利用垃圾回收器自动释放掉不再使用的内存，从而减少了程序员的工作。代码访问安全性就是在 JIT 编译器将 IL 指令转换成本地代码的过程中，CLR 将执行代码验证过程，以确保代码是类型安全的（例如避免将一种类型转换成与其不兼容的类型或出现非法指针等情况）。

1.4 .NET Framework 类库

正如 C 语言有 C 标准库一样，.NET 为所有运行在 CLR 环境下的应用程序提供了一个功能强大的公用代码库，即.NET Framework 类库（Framework Class Library，FCL），类库中包含数量庞大的类，这些类是以面向对象理论为基础而设计的，几乎封装了操作系统的所有对外编程接口（Application Program Interface，API），提供创建各种应用程序所需的广泛功能，而且这种类库是针对.NET 平台而非特定的编程语言的，这就使得使用 C#、VB.NET 和 C++.NET 等.NET 编程语言的程序员以一致的编程方式解决了不同语言之间相互集成的问题。

从开发人员的角度来看，编写托管代码的最大好处是可以使用.NET 基类库。开发人员使用基类库的方法是简单地实例化它们、调用它们的方法或者开发扩展其功能的派生类。

.NET 基类是一个内容丰富的托管代码类集合，它可以完成以前要通过 Windows API 来完成的绝大多数任务。这些类派生自与中间语言相同的对象模型，也基于单一继承性。无论.NET 基类是否合适，都可以实例化对象，也可以从它们派生自己的类。

.NET Framework 类库（FCL）中的 7000 多种类型——类、结构、接口、枚举和委托，组成了.NET Framework 的核心部分。一些 FCL 的类包含了 100 多个方法、属性和其他成员。所以学习 FCL 并不是轻松的事情。本书将主要说明如何使用.NET 基类库中的各种类，即各种基类是如何工作的。.NET 基类主要包括：

- IL 提供的核心功能，例如通用类型系统中的基本数据类型。
- Windows GUI 支持和控件。
- Web 窗体（ASP.NET）。
- 数据访问（ADO.NET）。
- 目录访问。
- 文件系统和注册表访问。
- 网络和 Web 浏览。
- COM 互操作性。

附带说一下，根据 Microsoft 源文件，.NET 基类几乎完全是用 C#编写的。

1.5 命名空间

因为.NET Framework 类库中包含数千个类，所以程序设计人员需要以快捷的方法找到所需要的类。将这些类库分组到命名空间中，将功能相似的类组织在一起，如进行文件、目录存取的 File、Directory 类就分类到 System.IO 命名空间下。命名空间也是.NET 避免类名冲突的一种方式，例如有两个程序员 A 和 B，他们处于同一个程序开发小组中，程序员 A 开发了一个名为 BankCustomer 的类，程序员 B 也开发了一个名为 BankCustomer 的类，那么在进行代码合并的时候，对于应用程序所创建的 BankCustomer 类对象而言，就不知道其引用的是哪个程序员所开发的 BankCustomer 类的成员，这样就造成了类名的冲突。利用命名空间可以解决这个冲突，使程序员 A 所开发的 BankCustomer 类位于命名空间 A 下，程序员 B 所开发的 BankCustomer 类位于命名空间 B 下，这样在应用程序中就可以用 A.BankCustomer 和

B.BankCustomer 来明显区分这两个类了，如图 1-7 所示。

```
namespace A
{
    public class BankCustomer{}
}
```

```
namespace B
{
    public class BankCustomer{}
}
```

图 1-7　命名空间解决类名冲突

命名空间不过是数据类型的一种组合方式，但命名空间中所有数据类型的名称都会自动加上该命名空间的名字作为其前缀。命名空间还可以相互嵌套。例如大多数用于一般目的的.NET 基类位于命名空间 System 中，基类 Array 在这个命名空间中，所以其全名是 System.Array。

.NET 需要在命名空间中定义所有的类型，例如可以把 Customer 类放在命名空间 YourCompanyName 中，则这个类的全名就是 YourCompanyName. Customer。

注意：如果没有显式提供命名空间，类型就添加到一个没有名称的全局命名空间中。

Microsoft 建议在大多数情况下都至少要提供两个嵌套的命名空间名，第一个是公司名，第二个是技术名称或软件包的名称，而类是其中的一个成员，例如 YourCompanyName.Sales Services.Customer。在大多数情况下，这么做可以保证类的名称不会与其他组织编写的类名冲突。

1.6　ADO.NET——数据和 XML

ADO.NET 是新一代的 ADO 技术，它为非连接的编程提供了更好的支持，而且在 System.Xml 命名空间中提供了丰富的 XML 支持，如图 1-8 所示。

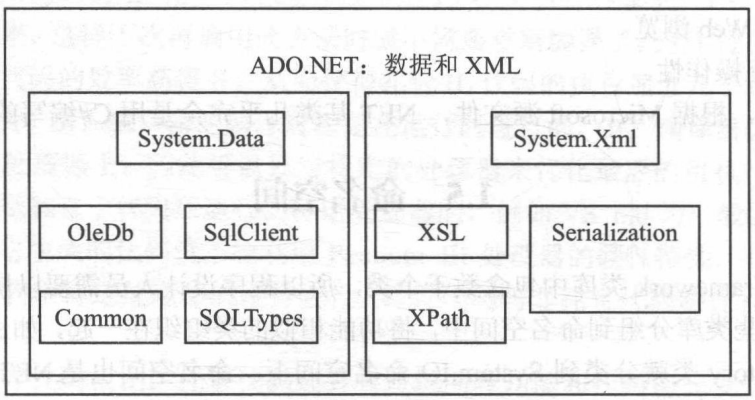

图 1-8　ADO.NET

System.Data 命名空间包含组成 ADO.NET 对象模型的类。粗略地划分，ADO.NET 对象模型被分为两层：链接层和非链接层。

System.Data 命名空间包含了 DataSet 类，它描述了多个表以及表之间的关系。这些数据集是完全自包含的数据结构，它们可以由多种数据源来填充。其中一种数据源是 XML；另一

种是 OLE DB；第三种是 Microsoft SQL Server 直连适配器。

System.Xml 命名空间提供了 XML 的支持。它包含了符合 W3C 标准的 XML 分析器和编写器。System.Xml.Xsl 命名空间提供了可扩展样式表语言转换。Xpath 的实现支持了在 XML 中数据图结构的导航。System.Xml.Serialization 命名空间为 XML Web Service 提供了完整的核心基础结构。

1.7 XML Web Service

XML Web Service 是.NET 平台的核心部分。它们是不同 Web 应用程序之间相互提供、使用数据与功能的基础机制，这些 Web 应用程序分布在单位内部或不同的单位之中。

XML Web Service 是为其他应用程序提供数据和服务的应用程序逻辑单元。应用程序可以通过工业标准 Web 协议和数据格式来访问 XML Web Service，如 HTTP、XML 和简单对象访问协议（Simple Object Protocol，SOAP），而不用考虑每个 XML Web Service 是如何实现的，如图 1-9 所示。

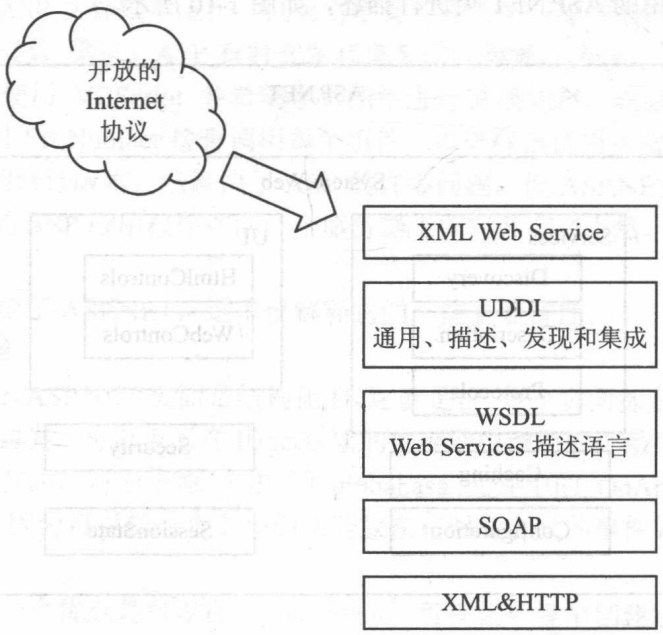

图 1-9　XML Web Service

1. XML 和 HTTP

XML Web Service 是使用 XML 和 HTTP 构建的，所以 XML Web Service 的运行可以不受防火墙的限制。另外，因为 XML 和 HTTP 是工业标准，所以任何支持 XML 和 HTTP 的平台都可以和 XML Web Service 协同工作。

2. SOAP

SOAP 定义了在与 XML Web Service 协同工作时如何格式化、发送和接收消息。SOAP 也是建立在 XML 和 HTTP 上的工业标准，所以任何支持 SOAP 的平台都可以支持 XML Web Service。

3. Web 服务描述语言

Web 服务描述语言（Web Services Description Language，WSDL）是一种 XML 格式，用来描述服务器提供的网络服务。可以使用 WSDL 创建一个文件，用来指明服务器提供的服务以及服务器支持的每个服务的操作集。

4. 通用、描述、发现和集成

通用、描述、发现和集成（Universal Discovery Description and Integration，UDDI）是注册和查找 XML Web Service 的工业标准。通过使用 UDDI，开发人员可以发现并使用在 Internet 上已经公开的可用的 XML Web Service。

1.8 Web 窗体和服务

ASP.NET 是建立在公共语言运行库之上的编程框架，公共语言运行库可用于在服务器上构建功能强大的 Web 应用程序。ASP.NET Web 窗体提供了易用且功能强大的方法来构建动态的 Web UI 页面。ASP.NET 的 XML Web Service 为构建分布式 Web 应用程序提供了构建模块。下面对一些比较常用的 ASP.NET 类进行描述，如图 1-10 所示。

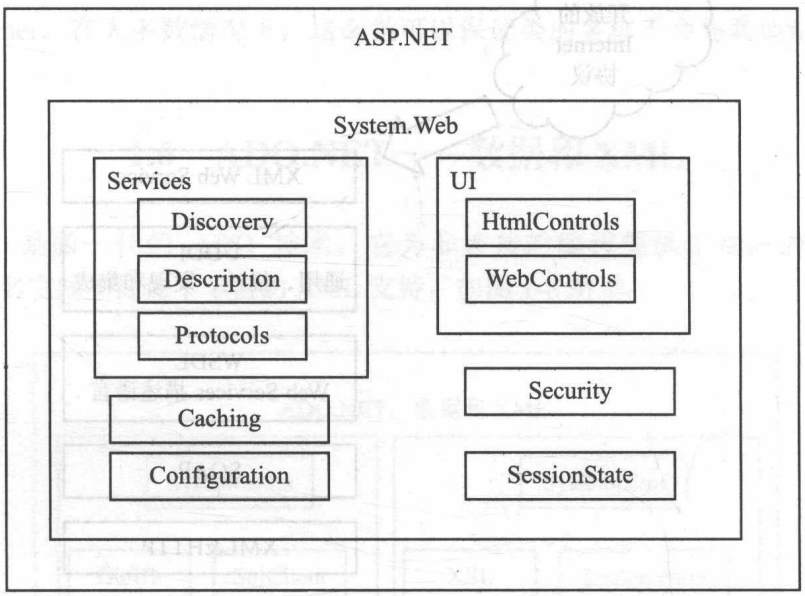

图 1-10 ASP.NET

1. System.Web

在 System.Web 命名空间中，有一些更底层的服务，如缓存、安全性和配置等，它们可以在 XML Web Service 和 Web UI 之间进行共享。

2. System.Web.Services

System.Web.Services 命名空间提供处理 XML Web Service 的类。

3. 控件

有两种类型的控件：HTML 控件和 Web 控件。System.Web.UI.HtmlControls 命名空间提供对 HTML 标记的直接映射，System.Web.UI. WebControls 命名空间能够使用模板来组织控件。