郭郑州 崔群法 张银鹤 编著

ASP.NET 3.5

从入门到精通

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

内 容 简 介

本书注重从初学者的认识规律出发，强调实用性和可操作性。对ASP.NET 3.5网站开发的基本概念和基本设计方法的讲解浅显易懂、深入浅出，并且安排了大量典型实用的例子，使学习者结合实践，掌握设计的方法和技巧。

本书内容包括：ASP.NET 3.5的环境配置、C#3.5编程基础、ASP.NET 3.5基本对象、Web.config文件、控件的应用、访问各种外部数据（XML、文件和数据库）。还介绍了开发高级ASP.NET网站所需掌握的技术，比如LINQ和WCF等，此外还包括ASP.NET 3.5新增Ajax技术的内容。

图书在版编目（CIP）数据

ASP.NET 3.5从入门到精通/郭郑州，崔群法，张银鹤编著.—北京：电子工业出版社，2009.2
ISBN 978-7-121-07832-3

Ⅰ. A… Ⅱ. ①郭… ②崔… ③张… Ⅲ. 主页制作—程序设计 Ⅳ. TP393.092

中国版本图书馆CIP数据核字（2008）第181673号

责任编辑：姜 影
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：32.875 字数：840千字
印 次：2009年2月第1次印刷
定 价：58.00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

ASP.NET是一个统一的Web平台，可提供生成企业级应用程序必需的所有服务。ASP.NET在.NET Framework基础上运行，因此所有.NET Framework功能都适用于ASP.NET应用程序，特别是ASP.NET 3.5技术提高了网络系统平台开发的效率和安全性，如新增LINQ和WCF、集成ASP.NET Ajax、增强的数据控件等。要进行ASP.NET 3.5开发Web应用程序，使用Visual Studio 2008和C#是最佳选择，也深受广大编程人员的青睐。

第1章以最新的.NET Framework 3.5版本为例向读者介绍什么是.NET Framework及其重要组成部分、.NET Framework 3.5的新增特性，还包括如何配置.NET Framework的开发环境及使用C#创建.NET应用程序。

第2章主要介绍C#的语法，包括声明C#中的常量、变量、运算符、表达式、枚举和面向对象设计等，还对C# 3.5中的语言新增功能进行了介绍，如扩展方法、匿名类型和Lambda表达式等。

第3章主要介绍制作Web所必须掌握的一些控件，包括HTML控件、文本控件、按钮控件、选择控件、日历控件及Panel和FileUpload控件等。

第4章主要介绍ASP.NET 3.5中验证控件、母版和内容页控件、向导Wizard和广告AdRotator控件的使用。

第5章主要介绍如何在ASP.NET中创建用户控件和自定义控件，内容包括用户控件的概述、创建用户控件和自定义控件、定义用户控件和自定义控件的属性及添加方法等。

第6章详细介绍如何配置ASP.NET应用程序，以及应用程序配置文件Web.config的结构。

第7章主要介绍如何使用GDI+向Web窗体绘制图形图像，以及在ASP.NET页面中打开、保存不同类型的文件和文件的读写操作等知识。

第8章主要介绍如何处理XML，包括XML语法格式、实现XML数据绑定并显示、XML的转换、创建与读取XML的方法。

第9章主要介绍ADO.NET，内容包括ADO.NET的组成、ADO.NET对象模型中的组件和事务的处理。

第10章主要介绍如何将数据绑定到控件显示，包括SqlDataSource数据源控件、数据绑定语法和绑定控件的使用，其中重点介绍新增数据绑定控件ListView的使用。

第11章主要介绍ASP.NET的数据缓存，包括缓存机制的概述、使用ASP.NET的页面缓存和应用程序缓存。

第12章详细介绍了ASP.NET 3.5中新增的数据访问方式LINQ，包括LINQ概述、LINQ查询表达式、查询操作、LinqDataSource数据源控件及LINQ To SQL等。

第13章主要介绍使用WebPart控件制作可自定义个性化设置的Web，包括WebPart控件概述、使用以及Web控件间的通信等。

第14章主要介绍如何使用ASP.NET内置的安全机制构建登录，包括身份验证概述、成员

角色和登录控件简介、创建ASPNETDB.MDF数据库和使用角色提供模型等。

第15章主要介绍使用ASP.NET 3.5开发Ajax网站要掌握的技术，包括部署ASP.NET Ajax环境、基本控件的作用及常用特效的实现方法等。

第16章介绍了3种高级开发技术，分别是使用Web服务、WCF和Crystal Report报表。

第17章主要讲解企业网站的一个模块——文件管理系统，该模块主要实现了一个文件管理器，可以对文件和文件夹进行各种管理，包括上传、添加、重命名和删除等。

第18章主要讲解企业网站的另一个模块——内容管理系统，该模块采用了三层架构来实现。内容包括：模块的需求分析、整体架构、数据库及通用功能的设计，还对每一层的实现进行详细介绍。

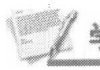
参加本书编写与制作的作者除封面署名者以外，还有于利敏、董志鹏、赵俊昌、秦雨、李振、王俊伟、唐有明、王咏梅、郑千忠、朱俊成、孙宇霞、郝春雨、王伟平、陈军红、张水波等。在此，编者对他们表示衷心的感谢。由于编写时间仓促，作者水平有限，书中难免会有错误和疏漏之处，恳请广大读者批评和指正。

目　录

第1章

.NET Framework体系结构

Microsoft发布的.NET Framework简称为.NET，是支持生成和运行下一代应用程序和Web服务的内部Windows组件，提供托管执行环境，简化开发和部署以及与各种编程语言的集成。本章以最新的.NET Framework 3.5版本为例，向读者介绍什么是.NET Framework及其重要组成部分，.NET Framework 3.5的新增特性，如何配置.NET Framework的开发环境及使用C#创建.NET应用程序。



- ▲ 了解C#语言
- ▲ 理解.NET Framework概念
- ▲ 理解并熟悉公共语言运行时的概念及组成
- ▲ 熟悉CTS和CLS
- ▲ 理解中间语言的概述
- ▲ 了解CLR的执行过程和内存管理机制
- ▲ 熟悉.NET Framework类库的结构
- ▲ 了解.NET Framework中程序集的概念
- ▲ 掌握CLR命名空间的定义及引用方法
- ▲ 掌握.NET环境的配置方法
- ▲ 了解各种.NET应用程序的概念

1.1 .NET Framework与C#

C#是随.NET Framework一起发布的一种新语言，是一种崭新的面向对象的编程语言，强调以组件为基础的软件开发。不但结合了Visual Basic的简单易用性，同时也提供了Java和C++语言的灵活性和强大功能。C#在.NET Framework中扮演着一个重要角色，可以说是Microsoft公司面向下一代互联网软件和服务战略的重要内容，也是编写.NET Framework应用程序的首选。

1.1.1 C#概述

C#是用于创建运行在.NET公共语言运行时上的应用程序的语言之一，是Microsoft专门为使用.NET平台而创建的，从C语言和C++语言演化而来，并且结合了其他语言的许多优点，例如Visual Basic的易用性。

C#本身是面向对象的语言，进一步提供了对面向组件（Component Oriented）编程的支持。现代软件设计日益依赖于包含和描述功能包形式的软件组件。这种组件关键在于通过属性（Property）、方法（Method）和事件（Event）来提供编程模型，以及关于组件声明信息的属性（Attribute）；C#提供的语言构造直接支持这些概述，这使得C#语言自然而然成为创建和使用软件组件的首选。

C#具有几个非常优秀的用于构造健壮和持久应用程序的特性。

- 垃圾回收自动回收不再使用的对象所占用的内存。
- 异常处理提供了结构化的错误检测和恢复方法。
- 类型安全的语言设计则避免了读取未初始化的变量、数组索引超出边界或执行未经检查的类型强制转换等情形。

此外，C#还具有一个统一的类型系统，所有C#类型（包括像int和string之类的基础数据类型）都继承于一个唯一的基类型：Object。因此，所有类型都共享一组通用操作，并且任何类型的值都能够以一致的方式进行存储、传递和操作。另外，C#同时支持用户定义的引用类型和值类型，既允许对象的动态分析，也允许轻量结构的内联存储。

为了确保C#程序和库能够以兼容的方式逐步演进，C#的设计中充分强调了版本控制。许多语言都不太重视这一点，导致采用这些语言编写的程序，常常因为其所依赖的库的更新而无法正常工作。C#的设计在某些方面直接考虑到了版本控制的需要，其中包括单独使用的virtual和override修改、方法重载决定规则以及对显式接口成员声明的支持。

总之，C#是一个易于使用的，能够开发出功能强大、安全、稳定的应用程序的语言，在本书的第2章中将详细描述C#的这些特性。

1.1.2 .NET Framework 3.5概述

.NET Framework是支持生成和运行下一代应用程序和XML Web服务的内部Windows组件。.NET Framework旨在实现下列目标。

- 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，还是在本地执行而在Internet上分布，或者是在远程执行。
- 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。
- 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。
- 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。
- 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。
- 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。

1. .NET Framework组件

.NET Framework主要有两个组件：公共语言运行时和.NET Framework类库。公共语言运行时是.NET Framework的基础。读者可以将运行时看作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，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。事实上，代码管理的概念是运行时的基本原则。以运行时为目标的代码称为托管代码，不以运行时为目标的代码称为非托管代码。

.NET Framework的另一个主要组件类库，是一个综合性的面向对象的可重用类型集合。读者可以使用它开发多种应用程序，这些应用程序包括传统的命令行或图形用户界面（GUI）应用程序，也包括基于ASP.NET所提供的最新的应用程序（如Web窗体和XML Web服务）。

如图1-1所示的.NET Framework平台上显示了公共语言运行时、类库与应用程序以及整个系统之间的关系。

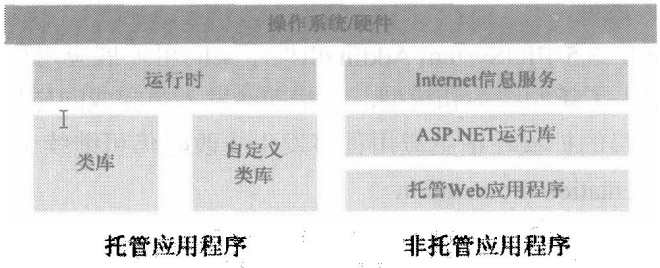


图1-1 .NET Framework平台

2. .NET Framework 3.5简介

.NET Framework 3.5以.NET Framework 2.0和.NET Framework 3.0为基础，包括.NET Framework 2.0和3.0的Service Pack。主要包括如下的组件。

- .NET Framework 2.0。
- .NET Framework 2.0 Service Pack 1，更新包含在.NET Framework 2.0中的程序集。
- .NET Framework 3.0，使用.NET Framework 2.0或.NET Framework 2.0 Service Pack 1（如果已安装）中存在的程序集，并且包含.NET Framework 3.0中引入的技术所必需的程序集。例如，Windows Presentation Foundation（WPF）所必需的PresentationFramework.dll和PresentationCore.dll就随.NET Framework 3.0一起安装。
- .NET Framework 3.0 Service Pack 1，更新在.NET Framework 3.0中引入的程序集。
- 一些新程序集，它们为.NET Framework 2.0和3.0提供附加功能，同时还提供.NET Framework 3.5中新采用的技术。

如果在计算机上安装 .NET Framework 3.5时缺少上述任何组件，那么会自动将安装它们。应用程序无论针对的是.NET Framework 2.0、3.0还是3.5，都使用相同的程序集。例如，对于使用WPF并针对.NET Framework 3.0的应用程序，其所使用的mscorlib程序集实例与使用Windows窗体并针对.NET Framework 2.0的应用程序是相同的。如果.NET Framework 2.0 Service Pack 1已安装在计算机上，那么mscorlib.dll已更新，并且两个应用程序将都使用mscorlib.dll的更新版本。

.NET Framework 2.0、3.0和3.5之间的关系不同于1.0、1.1和2.0之间的关系。.NET Framework 1.0、1.1和2.0是彼此完全独立的，对于其中任何一个版本来说，无论计算机上是否存在其他版本，都可以存在于该计算机上。当1.0、1.1和2.0位于同一台计算机上时，每个版本都有自己的公共语言运行时、类库和编译器，等等。应用程序可以选择是针对1.0、1.1还是2.0。

3. .NET Framework 3.5重要新功能

.NET Framework 3.5为2.0和3.0中的技术引入了新功能，并以新程序集的形式引入了其他技术。下列技术是随.NET Framework 3.5引入的技术。

• LINQ

LINQ（Language Integrate Query，语言集成查询）是Visual Studio 2008和.NET Framework 3.5中的新功能。LINQ将强大的查询功能扩展到C#和Visual Basic的语言语法中，并采用标准的、易于学习的查询模式。可以对此技术进行扩展以支持几乎任何类型的数据存储。

• 外接程序和扩展性

.NET Framework 3.5中的System.AddIn.dll程序集向可扩展应用程序的开发人员提供了强大而灵活的支持。引入了新的结构和模型，可帮助开发人员完成向应用程序添加扩展性的初始工作，并确保开发人员的扩展在宿主应用程序发生更改时仍可继续工作。

• Windows Presentation Foundation

在.NET Framework 3.5中，Windows Presentation Foundation包含多个方面的更改和改进，其中包括版本控制、应用程序模型、数据绑定、控件、文档、批注和三维UI元素。

• WCF和ASP.NET的Ajax集成

WCF与ASP.NET中的异步JavaScript和XML（Ajax）功能的集成提供了一个端对端的编程模型，可用于构建可以使用WCF服务的Web应用程序。在Ajax样式的Web应用程序中，客户端（例如Web应用程序中的浏览器）通过使用异步请求来与服务器交换少量的数据。在ASP.NET中集成Ajax功能可提供一种生成WCF Web服务的简单方法，通过使用浏览器中的客户端JavaScript可以访问这些服务。

• ClickOnce清单新增了一些密码类，用于验证和获取有关ClickOnce应用程序的清单签名的信息

在这里仅列举了.NET Framework 3.5中的重要新功能和新特性，但不是全部。读者如果需要了解更多，可到网站http://www.microsoft.com上查找。

1.1.3 ASP.NET 3.5概述

ASP.NET作为.NET Framework的一部分，是一种可以在高度分布的Internet环境中简化应用程序开发的平台，提供了建立和部署企业级Web应用程序所必需的服务。ASP.NET 3.5为能够面向任何浏览器或设备的更安全的、更强可升级性的、更稳定的应用程序提供新的编程模型和基础结构。

.NET Framework 3.5针对ASP.NET 3.5和Visual Web Developer中的特定方面提供了增强功能。最重要的改进在于支持Ajax的网站开发。ASP.NET 3.5支持使用一组新的服务器控件和API进行以服务器为中心的Ajax开发。通过添加ScriptManager控件和UpdatePanel控件，可以让现有ASP.NET 2.0页面支持Ajax功能，这样更新页面时将无需整页刷新。以下主要对三个方面的改进进行介绍：

• ASP.NET Ajax和Visual Web Developer改进

ASP.NET 3.5还支持使用名为Microsoft Ajax Library的新客户端库进行以客户端为中心的Ajax开发。Microsoft Ajax Library支持以客户端为中心、面向对象且独立于浏览器的开发。借

助ECMAScript（JavaScript）中的库类，可以提供丰富的UI行为，而无需反复访问服务器。开发人员可以根据应用程序的需要，调整以服务器为中心和以客户端为中心这两种开发模式的比例。此外，Visual Web Developer还包括改进的对JavaScript的IntelliSense支持和对Microsoft Ajax Library的支持。

现在，ASP.NET 3.5和Visual Web Developer支持创建基于ASMX和WCF的Web服务，还支持在采用Microsoft Ajax Library的网页中无缝使用任意实现。此外，Forms身份验证、角色管理和配置文件的服务器端应用程序服务现已作为Web服务公开，这些服务可以在WCF兼容应用程序（包括客户端脚本和Window窗体客户端）中使用。ASP.NET允许所有的基于Web的应用程序共享这些公共应用程序服务。

· ASP.NET 3.5的其他改进

ASP.NET 3.5中的其他改进包括：用于显示数据的新数据控件ListView，通过ASP.NET数据源控件结构向Web开发人员公开语言集成查询（LINQ）的新数据源控件LinqDataSource，用于合并预编译程序集的新工具ASP.NET合并工具（Aspnet_merge.exe），与IIS 7.0的紧密集成。ListView是一个可高度自定义的控件（使用模板和样式），该控件还支持编辑、插入和删除操作以及排序和分页功能。一个名为DataPager的新控件为ListView提供了分页功能。可以使用合并工具来合并程序集以支持各种部署和发布管理方案。ASP.NET 3.5和IIS 7.0的集成提供了对任何内容类型使用ASP.NET服务（如身份验证和缓存）的能力，还提供了使用ASP.NET托管代码开发服务器管理模块的能力，并且支持模块和处理程序的统一配置。

· Visual Web Developer的其他改进

Visual Web Developer中的其他改进包括：多目标支持、包含Web应用程序项目、新的“设计”视图、新的级联样式表（CSS）设计工具以及对LINQ for SQL数据库的支持。多目标功能使用户能够使用Visual Web Developer针对特定的.NET Framework 版本（包括2.0、3.0和3.5版）开发Web应用程序。

1.1.4 公共语言运行时（CLR）

.NET Framework提供一个称为公共语言运行时（Common Language Runtime）的运行环境，运行代码并提供使开发过程更轻松的服务。作为.NET Framework的核心组件，它是执行时管理代码的代理，提供内存管理、线程管理和远程处理等核心服务。如图1-2所示为公共语言运行时环境中的各个部分及其提供的重要服务。

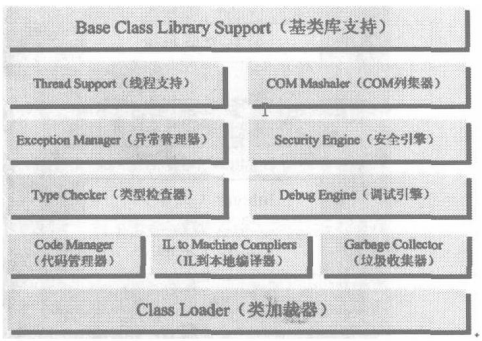


图1-2 公共语言运行时环境

公共语言运行时通过公共类型系统（Common Type System，CTS）和公共语言规范（Common Language Specification，CLS）定义了标准数据类型和语言间互操作性的规则。Just-In-Time编辑器在运行应用程序之前把中间语言（Intermediate Language，IL）代码转换为可执行代码。CLR还管理应用程序，在应用程序运行时为其分配内存和解除分配内存。

1.1.5 .NET Framework类库

.NET Framework类库是生成.NET应用程序、组件和控件的基础。.NET Framework包括的类可执行下列功能：

- 表示基础数据类型和异常
- 封装数据结构
- 执行I/O
- 访问关于加载类型的信息
- 调用.NET Framework安全检查
- 提供数据访问、多客户端GUI和服务器控制的客户端GUI

.NET Framework类库是一个由.NET Framework SDK中包含的类、接口和值类型组成的库，提供了对系统功能的访问，是建立.NET Framework应用程序、组件和控件的基础。.NET Framework类库中还包含了.NET Framework中定义的所有类，如图1-3所示为.NET Framework类库与.NET Framework之间的关系，以及一些.NET Framework类库中的成员。

类通过继承从其他类创建。通过继承，一个类可以访问另一个类定义的方法和属性。另外，除了继承一个类的属性和方法之外，还可以修改已有方法的动作或者属性的行为，这称为重写（Overriding）。

.NET Framework中的所有类和用户创建的类都组织成层次结构，.NET Framework层次结构的基本类为System.Object，也就是说System.Object类位于层次结构的最顶端，称为超类（Super Class），提供了.NET Framework中所有类的基本功能。如图1-4所示为.NET Framework类库中定义的一些类之间的关系。

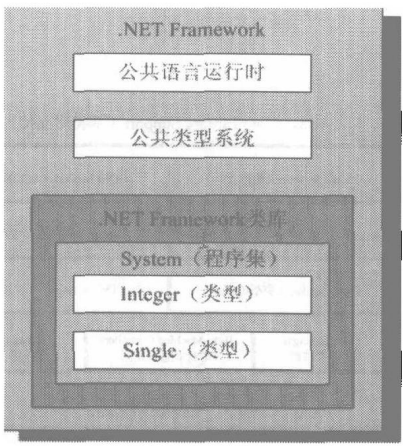


图1-3 .NET Framework类库与.NET Framework的关系

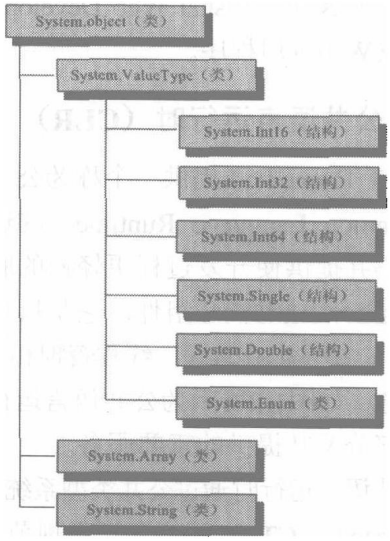


图1-4 .NET Framework类库层次结构

在图1-4中，Int16、Int32、Int64、Single、Double结构都派生自System.ValueType，System.Enum类也派生自System.ValueType，而System.ValueType派生自System.Object。Array和String类都直接派生自System.Object。这说明不管是什么类都位于.NET Framework类库层次结

构的某个位置，最终都派生自基类System.Object，这也是理解.NET Framework类库的关键。

1.1.6 程序集

程序集（Assembly）是.NET Framework的生成块，它们构成了部署、版本控制、重复使用、激活范围控制和安全权限等基本单元。程序集可以是静态的，也可以是动态的。静态程序集可以包括.NET Framework类型（接口和类），以及该程序集的资源（位图、JPEG文件、资源文件等），存储在磁盘上的可移植可执行文件中。还可以使用.NET Framework来创建动态程序集，直接从内存运行并且在执行前不存储到磁盘上，在执行动态程序集后可以将它们保存在磁盘上。

.NET Framework类库由许多程序集组成，用于读取和写入文件，从数据库保存和检索信息，以及提供窗体的功能。如表1-1所示为.NET Framework类库的部分程序集及其作用。

表1-1 常见程序集

名称	说明
System.dll	该程序集用于定义主要数据类型，如Integer和Long，另外还定义了最基本的类，如System.Object
System.Windows.Forms.dll	该程序集包含用来实现桌面应用程序使用的窗体的组件，以及创建这些窗体的控件
System.XML.dll	该程序集包含处理XML文档所必需的组件，XML是ADO.NET与Internet相关服务的主要的传输协议
System.Drawing.dll	该程序集包含用于向输出设置（如屏幕、打印机等）绘制各种图形（直组、圆形等）的组件
System.Data.dll	该程序集用于定义组成ADO.NET的组件，而ADO.NET提供了Visual Studio.NET使用的数据库处理服务

1. 程序集内容

通常程序集的内容由4个元素组成：程序集清单，包括程序集元数据；类型元数据；实现这些类型的MSIL代码；资源集。其中只有程序集清单是必需的，当然也需要类型或资源来向程序集提供任何有意义的功能。程序集中的这些元素有几种分组方法。开发人员可以将所有元素分组到单个物理文件中，如图1-5所示为单文件程序集的结构。

在如图1-6所示的多文件程序集结构中，假设应用程序的开发人员已选择将一些实用工具代码单独放入另一个模块中，同时在其源文件中保留一个较大的资源文件（在此例中为一个.bmp图像）。.NET Framework只在文件被引用时下载该文件，通过将很少引用的代码保留在独立于应用程序的文件中来优化代码下载。

在图1-6中，所有的文件均属于一个程序集，如MyAssembly.dll所包含的程序集清单文件中所述。对于该文件系统，这三个文件是三个独立的文件。但是要注意，文件Util.netmodule被编译为一个模块，因为它不包含任何程序集信息。在创建了程序集后，该程序集清单就被添加到MyAssembly.dll，指示程序集与Util.net模块和Graphic.bmp的关系。

2. 程序集清单

每一程序集，无论是静态的还是动态的，均包含描述该程序集中各元素彼此如何关联的数

据集合。程序集清单就包含这些程序集元数据。程序集清单包含指定该程序集的版本要求和安全标识所需的所有元数据，以及定义该程序集的范围和解析对资源和类的引用所需的全部元数据。程序集清单可以存储在具有MSIL代码的PE文件（.exe或.dll）中，也可存储在只包含程序集清单信息的独立PE文件中，如图1-7所示。

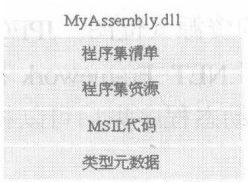


图1-5　单文件程序集

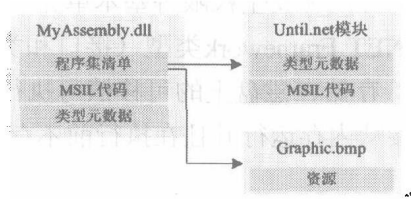


图1-6　多文件程序集

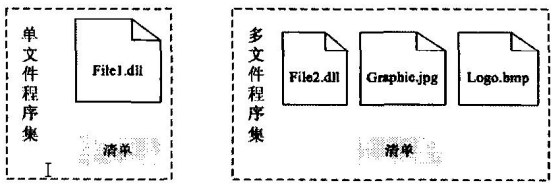


图1-7　单文件和多文件程序集清单

对于有一个关联文件的程序集，该清单将被合并到PE文件中以构成单文件程序集。可以创建有独立清单文件的多文件程序集，也可以创建清单文件被合并到同一多文件程序集中某一PE文件的多文件程序集。

表1-2显示了在程序集清单中包含的信息，其中前4项（程序集名称、版本号、区域性和强名称信息）构成了程序集的标识。

表1-2　程序集清单信息

信息	说明
程序集名称	指定程序集名称的文本字符串
版本号	主版本号和次版本号，以及修订号和内部版本号。公共语言运行时使用这些编号来强制实施版本策略
区域性	有关该程序集支持的区域性或语言的信息。此信息只应用于将一个程序集指定为包含特定区域性或特定语言信息的附属程序集。（具有区域性信息的程序集被自动假定为附属程序集）
强名称信息	若已经为程序集提供了一个强名称，则为来自发行者的公钥
程序集中所有文件的列表	在程序集中包含每一文件的散列及文件名。请注意，构成程序集的所有文件所在的目录必须是包含该程序集清单的文件所在的目录
类引用信息	运行库用来将类引用映射到包含其声明和实现的文件的信息。该信息用于从程序集导出的类型
有关被引用程序集的信息	该程序集静态引用的其他程序集的列表。若依赖的程序集具有强名称，则每一引用均包括该依赖程序集的名称、程序集元数据（版本、区域性、操作系统等）和公钥

1.1.7 命名空间

命名空间提供了一个组织相关类和其他类的方式。与文件或者组件不同，命名空间是一种逻辑组合，而不是物理组合。不在同一个文件中的多个类可以共同包含在一个命名空间中，这样就创建了一个逻辑结构。

程序集是.NET Framework的构造块，一个物理程序集包含一个或者多个命名空间。例如System和System.IO命名空间都保存在System.dll程序集中。而一个命名空间也可能保存在两个程序集中。一个命名空间又包含一个或多个类型（Type）。在.NET Framework中，Integer或者String被认为是类型。每一个类也被视为类型。例如，System命名空间包含一个名为Int16的类型，它代表一个值。

1. 命名空间组成

命名空间还包含类、委托、结构、枚举和接口，它们都是类型。这些类型既有值类型又有引用类型，如下所示为这些类型的简单介绍。

• 类

它是引用类型，表示具有某个类类型的变量保存了内存地址。该内存地址指向分配给实际对象的内存。比如，工具箱中的控件都是类，窗体也是类。

• 委托

它也是引用类型，委托提供了一种机制让程序向Windows操作系统注册事件，以便Windows在运行时调用这些事件，进而调用合适的事件处理程序。委托（事件处理程序）也可以在运行时动态创建。

• 结构

它是值类型，表示数据直接保存在变量中。结构可以拥有属性和方法。主要的数据类型，如Int（Int16）或者Single在.NET Framework中都是结构。

• 枚举

它是一种受限形式的值类型。枚举没有属性和方法，而有域。枚举提供了一种机制把帮助记忆的名称与一个常量值相关联。因此，从概念上讲，枚举类似于常量。许多控件属性都是枚举值。

• 接口

定义了实现该接口的其他类必须支持的成员（方法、属性等）。也就是说，接口定义了类必须执行的任务，而不是类执行该任务的具体方式。例如，可以通过调用Array类的Sort方法排序数组。Sort方法使用接口IComparable或者IComparer的实现方式来确定如何排序数组中的元素。接口不会定义何种类型的对象排序，也不会定义对象如何排序。由用户创建的接口的实现方式定义如何排序对象。

2. 定义命名空间

要定义命名空间，就需要使用namespace关键字。namespace关键字用于声明一个范围。此命名空间范围允许用户组织代码并提供了创建全局唯一类型的方法，定义规则如下。

- 命名空间名可以是任何合法的标识符，命名空间名可以包含句号“.”。
- 即使未显式声明命名空间，也会创建默认命名空间。该未命名的命名空间（有时称为全局命名空间）存在于每一个文件中。全局命名空间中的任何标识符都可用于命名的命名空间中。