.NET KUANGJIA CHENGXU KAIFA YU YINGYONG

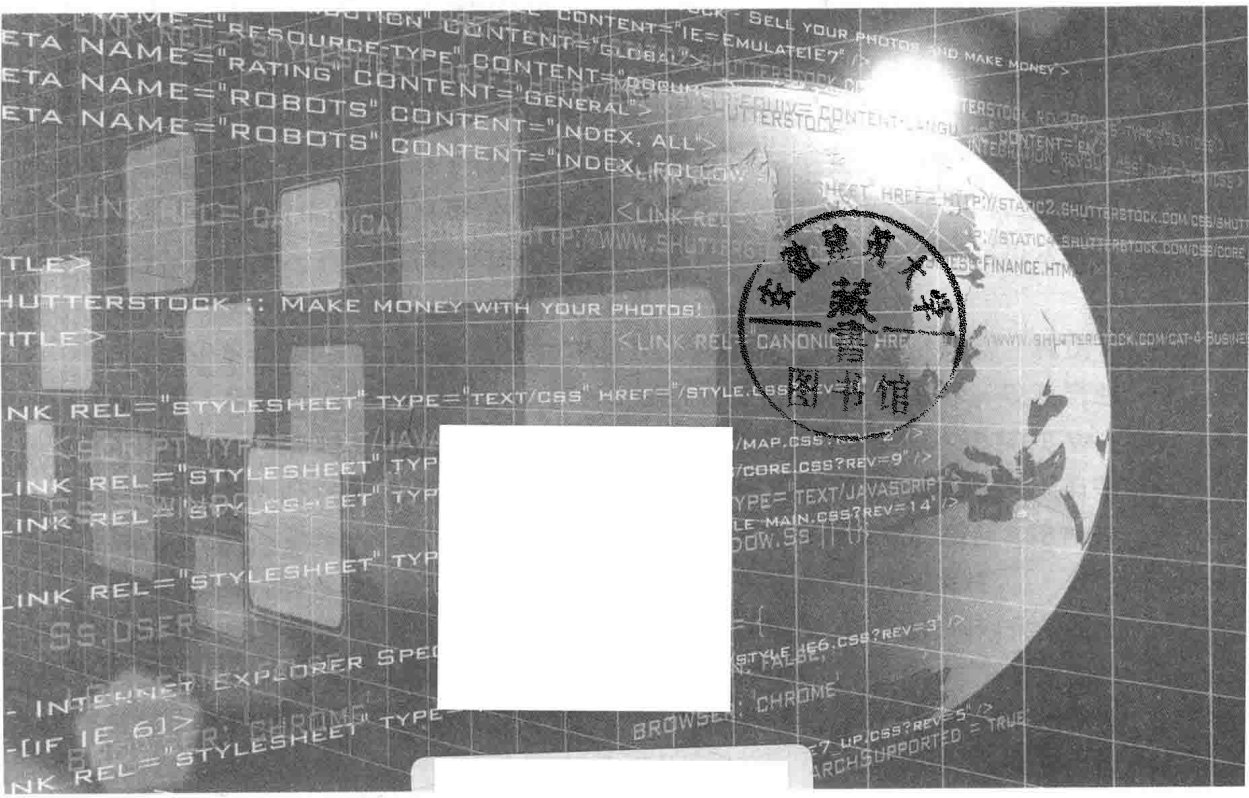
.NET 框架程序开发与应用

◎张永财 著

NORTHEAST NORMAL UNIVERSITY PRESS
WWW.NENUP.COM
东北师范大学出版社

.NET 框架程序开发与应用

▣ 张永财 著



NORTHEAST NORMAL UNIVERSITY PRESS
WWW.NSDUP.COM
东北师范大学出版社

图书在版编目(CIP)数据

.NET 框架程序开发与应用 / 张永财著. —— 长春：东北师范大学出版社，2017.5
ISBN 978-7-5681-3140-7

Ⅰ. ① N… Ⅱ. ①张… Ⅲ. ①互联网络－程序设计 Ⅳ. ① TP393.4

中国版本图书馆 CIP 数据核字（2017）第 133436 号

□策划编辑：王春彦
□责任编辑：卢永康　郎晓凯　□封面设计：优盛文化
□责任校对：赵忠玲　□责任印制：张允豪

东北师范大学出版社出版发行
长春市净月经济开发区金宝街 118 号（邮政编码：130117）
销售热线：0431-84568036
传真：0431-84568036
网址：http://www.nenup.com
电子函件：sdcbs@mail.jl.cn
河北优盛文化传播有限公司装帧排版
北京一鑫印务有限责任公司
2017 年 9 月第 1 版　2017 年 9 月第 1 次印刷
幅画尺寸：170mm×240mm　印张：17.25　字数：328 千

定价：60.00 元

前言

我国软件产业持续保持高速发展，已成为我国电子信息产业中增长最快、潜力最大的领域之一。IT 技术革命从第一代基于文本、第二代的基于图形，以及进入了基于 Web 的第三代，.NET 框架是微软在 2000 年专业开发者会议上提出的发展中的开发平台，这是一个革命性的应用程序开发平台。该平台建立在开发的 Internet 协议和标准上，采用了许多新的工具和服务，用于计算和服务，创建 XML Web 服务平台，该平台将信息、设备和人以一种统一的个性化的方式连接起来。

随着网络计算时代的到来，各种应用与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。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。.NET 框架是 C# 程序设计的一个提高，.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，.NET 底层框架技术可以从整体上把握 .NET 软件开发的方法。

ASP.NET MVC 是微软的一个 Web 开发框架，它整合了“模型 - 视图 - 控制器（MVC）”架构的高效和整洁、敏捷开发最新的思想与技术，以及当前 ASP.NET 平台的精华部分。ASP.NET MVC 可以完全替代传统的 ASP.NET Web Form，除了一些微不足道的 Web 小项目外，在各种 Web 开发项目中它都具有明显的优势。最新版 ASP.NET MVC 提供了高生产率的编程模型，结合 ASP.NET 的全部优势，组成更整洁的代码架构、测试驱动开发和强大的可扩展性。

在时代的背景下，我们有必要研究新的技术和新的知识，以适应时代的发展要求。本书通过分析总结框架设计的整体思路，框架内部的模式、框架灵活性的配置、框架工具的支持，并研究分析基于 .NET 平台的 web 开发，有助于了解框架设计的核心思想，快速掌握框架设计的技巧，并可以熟练地基于 .NET 平台进行软件开发。帮助大家从复杂、烦琐、无序的软件设计、开发和维护工作中解放出来，以良好的设计思维与架构方法为软件的开发和设计保驾护航。

在本书的编写中未列出的引用文献和论著，我们深表歉意，并同样表示感谢。

由于时间的仓促，编者水平有限，难免存在不足之处，在本书出版之际，我们真诚地希望读者对本书提出宝贵的意见和建议。

目录

第一章　.NET 平台与 .NET 框架

第一节　框架的概述

随着计算机技术的发展，软件的规模在不断扩大，复杂度在不断增加，软件危机也愈加明显地暴露出来。软件复用被认为是提高软件生产率、解决软件危机、提高软件质量和增强软件开放性的主要途径。框架是面向对象系统获得最大软件复用的方式之一，目前已有一些成熟的框架技术应用于相应的领域，帮助应用开发人员快速开发各种领域的应用软件，并提高软件的质量和可维护性。它们为企业应用框架的开发提供了借鉴和指导。

一、框架的定义

目前关于框架的定义较多，其中有以下 4 个比较有代表性的定义。

（1）Johnson 和 Foote 早在 20 世纪 80 年代末对框架的定义：框架是表现为解决一类特定问题的抽象设计的一组类。

（2）Firesmith 对框架的定义：框架是由一些相互协作的类组成的一个集合，它捕获了实现特定应用领域的公共需求、设计的主要机制以及小尺度的模式。

（3）Erich Gamma 等人认为：框架是构成一类特定软件可复用设计的一组相互协作的类。框架规定了应用的体系结构。它定义了整体结构类和对象的分割，各部分的主要责任，类和对象的协作机制，以及控制流程。框架预定义了这些设计参数，以便于应用设计者或实现者能集中精力于应用本身的特定细节。框架记录了其应用领域的共同的设计决策。因而框架更强调设计复用，尽管框架常包括具体的立即可用的子类。

（4）北京大学教授从构件的角度认为：框架由一组互相协作的构件组成，通过这些构件及其协作关系定义了应用系统的体系结构。这些成员构件通常是子框架、类树或类，大多以抽象的形式出现，实现细节放在具体子类中，构成了一个抽象设计，不同的具体子类可产生对设计的不同实现。框架作为构件使得用户可以复用设计，用户通过具体子类的嵌入而在框架中加入特殊功能。

综上所述，框架是一种软件重用技术，是一个应用软件系统的部分或整体的可重用设计，具体表现为一组抽象类以及其实例（对象）之间的相互作用方式。

二、框架的分类

（一）从组成范围分类

框架具有领域性，根据所面向的问题领域不同，框架主要分为三类：应用框架（Application Framework）、领域框架（Domain Framework）、支持框架（Support Framework）。

1. 应用框架

应用框架封装了各种专门的技术或功能并可应用于不同领域的应用开发。如ET++ 图形用户界面框架、MFC（Microsoft Foundation Classes，微软基本类库）、JFC（Java Foundation Classes，Java 基本类库），其中 JFC 是充分利用了面向对象技术进行设计开发的，并且运用了各种面向对象的设计模式。

2. 领域框架

领域框架为某个特定问题域的实现提供了专门的解决方案和功能，如各种生产控制系统框架、银行或报警系统框架、通信服务系统框架等。很多领域应用软件是为某个部门或厂家定制的，开发往往是从头开始，而领域框架的使用将最大限度地复用已有的设计经验与解决方案，降低开发成本，减少开发时间，并且可以提高系统的可靠性和可维护性等。

3. 支持框架

支持框架提供一些与计算机底层相关的特殊服务，如内存与文件系统的管理、设备驱动、分布式计算支持服务等。

这三类框架都可以为应用开发者提供相应的设计复用，减少代码的编写与维护工作。通常是开发某个特殊领域的面向对象框架，致力于提高领域应用的开发效率。

（二）从使用形式分类

为了开发灵活的、可复用的软件系统，面向对象设计方法主要运用两种基本的复用机制，即继承（Inheritance）和组合（Composition）。根据框架实际应用中运用这两种机制的差别，框架一般可分为以下三类：白盒（White-box）框架、黑盒（Black-box）框架和两者的混合框架。

白盒框架指的是主要通过类继承机制而应用于具体应用系统开发的那些框架，也称为体系结构驱动框架；那些主要通过框架中已有对象或构件的配置组合而应用于具体系统开发的框架则称为黑盒框架。白盒框架有较好的可扩展性与灵活性，可以方便地改变被复用的实现，但是它破坏了封装性，使用困难，框架用户必须知道框架的所有内部结构细节。黑盒框架使用简单，复用性好，易于实现运行时刻的动态绑定，而框架用户只需要了解框架的基本体系结构和热点，但是由于它隐藏了内

部细节，因而缺乏灵活性与可扩展性，并且这种框架的开发难度比较大。因此，框架在大多数情况下是白盒框架和黑盒框架两者的一种混合形式，同时提供类继承与对象组合这两种复用机制，以达到简单易用、灵活可扩展的目的。Jilles van Gurp 和 Jan Bosch 提出一种混合框架的概念结构，将框架中的元素分为白盒框架元素与黑盒框架元素两类，白盒框架元素表示框架中的抽象部分，包括各种接口与抽象类等元素，黑盒框架元素是各种继承于白盒框架中的元素并提供实现的各种构件和类。

（三）从使用范围分类

根据框架的使用范围可以将其分为系统结构框架（System Infrastructure Frameworks）、中间件集成框架（Middleware Integration Frameworks）和企业应用框架（Enterprise Application Frameworks）。

1. 系统结构框架用于简化开发可移植的、高效的系统体系结构，如操作系统框架、通信系统框架以及用户界面设计框架和语言处理工具等。

2. 中间件集成框架通常用来集成分布式应用程序和相关组件。它们可以增强软件开发人员开发模块化、可重用软件的能力，并且将软件无缝集成到分布式应用环境中。

3. 企业应用框架用于在某一特定企业应用领域中（如电信、电子设施、制造业和金融等）提供通用的业务控制或特定业务模式，是对该领域进行抽象而得出的，有助于减少实现应用的工作量，允许开发人员在涉及的领域开发高质量的软件，同时减少开拓市场的时间。

三、框架的优点

框架的开发的优点体现在 4 个方面：模块性（Modularity）、可重用性（Reusability）、扩展性（Extensibility）和反向控制（Inversion of Control）。

框架使用稳定的接口封装了具体的实现，从而增强了框架的模块性。框架使用这种机制将具体实现的影响和改变局部化，从而提高了软件产品的质量，局部化的好处在于减少了理解和维护具体实现的时间。

通过框架提供的稳定接口增强了软件产品的可重用性。当创建新的应用程序时，可以使用框架中定义的通用组件。这些通用组件由具有丰富领域知识和程序设计经验的程序员设计开发，它们满足一定应用领域的需求。使用通用组件避免了重新设计软件产品中这些部分带来的风险。可重用框架组件的使用也相应提高了程序员开发软件产品的效率，同时又保证了软件产品的质量和性能。

框架通过提供一种“钩子”方法提高它的扩展性，这些方法允许应用程序扩展框架稳定的接口。它们将稳定的接口和特定应用中可变的行为分离开来。框架的这种特性保证了能够方便地为新的应用提供新的服务和特性。

反向控制是框架运行时的一个特点。框架需要采用“注册 / 通知”机制处理应用程序流程，以满足框架能够处理领域中的所有应用的要求。首先，特定的应用应当向它使用的框架注册需要框架处理的程序流程部分；随后，在运行特定的应用程序后，某个已经注册了的程序流程运行条件得到满足时，框架使用它的派发机制通知应用程序执行该程序流程。反向控制保证了应用开发的简单性和扩展性。特定的应用只需要关注它需要实现的商业逻辑和触发商业逻辑的运行条件的开发，而将复杂的底层触发机制交给框架统一处理。

四、基于框架的应用开发

框架的出现改变了面向对象应用程序的传统开发过程，基于框架的应用开发过程主要包括 3 个阶段：领域分析、框架设计、框架实例化。

领域分析阶段主要利用应用开发经验，对典型的领域应用与领域相关的一些标准规范进行需求分析，包括框架的共性抽象和潜在的变化渴求分析。

框架设计阶段主要基于领域分析对框架的核心、公共部分和热点进行设计，并利用设计模式等面向对象技术协助框架设计以达到良好的复用性、灵活性和扩展性。

最后阶段是框架的实例化，这一阶段基于框架创建相应的具体应用程序。

五、企业应用框架模式

（一）框架模式和设计模式的关系

企业应用按框架来分类，包括 C++ 语言的 QT、MFC、GTK、Java 语言的 SSH、SSI，PHP 语言的 Smarty（MVC 模式），Python 语言的 Django（MTV 模式）等；按设计模式来分类，包括工厂模式、适配器模式、策略模式等；按框架模式来分类，包括 MVC、MTV、MVP、CBD、ORM 等。有很多程序员往往把框架模式和设计模式混淆，认为 MVC 是一种设计模式。实际上它们完全是不同的概念。框架与设计模式之间的不同主要表现以下 3 个方面。

1. 设计模式比框架更抽象。框架能够用代码表示，也能直接执行或复用；而设计模式是对在某种环境中反复出现的问题以及解决该问题的方案的描述，只有其实例才能表示为代码。

2. 设计模式是比框架更小的体系结构元素。一个典型的框架包括了多个设计模式。

3. 框架比设计模式更加特例化。框架总是针对某一特定的应用领域，一般的设计模式能被用于不同的应用领域。可以说，框架是软件，而设计模式是软件的知识。

设计模式是用来详细协助框架设计与文档编写的一种广为接受的技术，它的使用有利于提高框架的易理解性和可靠性。一个使用设计模式的框架比不使用设计模

式的框架更可能获得高层次的设计复用和代码复用，成熟的框架通常使用了多种设计模式。设计模式有助于获得无须重新设计就可适用于多种应用的框架体系结构。

框架主要由一系列设计模式和类（包括抽象类和具体类）组成，设计模式则是由一些类对象的共同参与而实现的，同一个类可以由几种不同的设计模式或框架所共享。框架中的这些重要元素间的关系，同时也说明了软件复用粒度按类、设计模式、框架的顺序依次递增。

（二）企业应用框架模式中的设计模式

目前大部分企业应用的框架模式分为标准 MVC 三层架构模式和企业根据项目有关需求结合企业自身情况（人员情况、技术情况、成本情况、进度效率情况等）自行设计的多层架构模式。

MVC 全名是 Model View Controller，是模型（Model）—视图（View）—控制器（Controller）的缩写，是一种软件设计典范，用一种业务逻辑和数据显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在界面和用户围绕数据的交互能被改进和个性化定制的同时而不需要新编写业务逻辑。MVC 被独特地发展起来用于将传统的输入、处理和输出功能映射到一个逻辑的图形化用户界面的结构中。

MVC 是一个框架模式，它强制性地使应用程序的输入、处理和输出分开。

1. 视图

视图是用户看到并与之交互的界面。对老式的 Web 应用程序来说，视图就是由 HTML 元素组成的界面，在新式的 Web 应用程序中，HTML 依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，它们包括 Adobe Flash 和 XHTML、XML/XSL、WML 等一些标识语言和 Web Services。

MVC 的好处是它能为应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个固定列表，作为视图来讲，它只是一种输出数据并允许用户操纵。

2. 模型

模型表示企业数据和业务规则。在 MVC 的 3 个部件中，模型拥有最多的处理任务。例如，它可能用 EJBs 和 ColdFusion Components 等构件对象来处理数据库，被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需要写一次就可以被多个视图重用，所以减少了代码的重复性。

3. 控制器

控制器接受用户的输入并调用模型和视图完成用户的需求，所以当单击 Web 页面中的超链接和发送 HTML 表单时，控制器本身不输出任何东西和做任何处理。它

只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。

六、.NET Framework 最新版重要技术解析

1. Windows Presentation Foundation

Windows Presentation Foundation（WPF）的核心是一个与分辨率无关、基于向量的显示引擎，它的主要目的在于充分利用现有硬件显示技术。WPF 为开发人员提供下一代显示系统，用于生成能带给用户震撼视觉体验的 Windows 客户端应用程序。为了更方便地使用 WPF 核心，微软公司还开发了一些外围扩展功能，这些功能包括可扩展应用程序标记语言（XAML）、控件、数据绑定、布局、二维和三维图形、动画、样式、模板、文档、媒体、文本和版式。

另外，WPF 作为 .NET 框架的一个独立组件发布，使开发人员能够生成融入了 .NET 框架类库的其他元素的应用程序。WPF 开发的客户端应用程序既可以运行在 Windows 操作系统下，也可以承载在浏览器之上。

2. Windows Communication Foundation

Windows Communication Foundation（WCF）的目的是为分布式计算提供可管理的方法，提供广泛的互操作性，并为服务定位提供直接的支持。WCF 通过一种面向服务的类型化编程模型简化关联应用程序的开发。通过提供分层的体系结构，WCF 支持多种风格的分布式应用程序开发。WCF 通道体系结构在底层提供了异步的非类型化消息传递基元，而建立在此基础之上的是用于进行安全可靠的事务处理数据交换的各种协议功能，以及广泛的传输协议和编码选择。

类型化编程模型（服务模型）设计用来降低分布式应用程序的开发难度，并为 WCF 开发人员提供熟悉的开发体验。该服务模型的特点在于它将 Web 服务的概念直接映射到 .NET 框架公共语言运行库（CLR）中的对应内容，包括将消息灵活且可扩展地映射到用 VisualC# 或 VisualBasic 等语言实现的服务。该服务模型提供支持松散耦合和版本管理的序列化功能，并提供与消息队列（MSMQ）、COM+、ASP.NET Web 服务、Web 服务增强（WSE）等 .NET 框架分布式系统技术以及其他功能的集成和互操作。

3. Windows Workflow Foundation

Windows Workflow Foundation（WWF）是一个框架，让开发人员可以在应用程序中创建系统或人工工作流。WWF 集编程模型、引擎、工具为一体，用于在 Windows 上快速生成启用工作流的应用程序。WWF 包括一个命名空间、一个进程内工作流引擎和多个 VisualStudio2010 设计器。WWF 既可用于解决简单方案，如根

据用户输入显示 UI 控件，也可用于解决大型企业遇到的复杂方案，如订单处理和库存控制。

WWF 提供了与其他 .NET Framework 技术（如 WCF 和 WPF）一致的熟悉的开发体验。WindowsWorkflowFoundationAPI 完全支持 VisualBasic.NET 和 C#、专用工作流编译器、在工作流中调试、图形工作流设计器，并支持完全用代码或标记开发工作流。WWF 还提供了可扩展模型和设计器，用于生成为最终用户或跨多个项目重用封装工作流功能的自定义活动。

4. 语言集成查询

语言集成查询（LINQ）在对象领域和数据领域之间架起了一座桥梁。传统上，针对数据的查询都是以简单的字符串表示，而没有编译时的类型检查或智能感知支持。此外，还必须针对以下各种数据源学习不同的查询语言：SQL 数据库、XML 文档、各种 Web 服务等。LINQ 使查询成为 C# 和 VisualBasic 中的一种语言构造。用户可以使用语言关键字和熟悉的运算符针对强类型化对象集合编写查询。

第二节　.NET 平台的概述

2000 年 6 月，微软发布了 .NET，期望能够做到“让任何人从任何地方，在任何时间使用任何设备访问 Internet 上的服务”。这是一种在软件开发、工程和应用方面广泛地支持 Internet 和 Web 的全新版本。为了更好地理解 .NET 平台，首先说一下 Internet 的发展。

所谓第一代 Internet，是以静态的页面呈现的，将数据或者文件编写成 HTML 页面并放置在 Internet 上共享。用户接口逻辑、业务逻辑与数据都存放在服务器上。应用系统在浏览器和 Web 服务的协助下运行。用户从客户端发出请求，服务器处理用户请求，产生静态 HTML 页面，然后在客户端显示结果。

第一代的 Internet 将应用程序的数据单独存放在后台数据库中，达到了真正的多人数据共享。但是这种结构仍有其固有的缺陷：首先是用户接口逻辑和业务逻辑都集中在一起，这样使业务逻辑与接口绑定，一旦业务逻辑变更，那么将不得不重新编译整个接口程序；其次，这种结构能够支持的客户端人数有限，一旦客户端人数增加执行效率就会下降。

第二代 Internet 属于 MicrosoftDNA（DistributedInterNetArchitecture）三层式的应用程序结构。应用程序包括三层：表现层、业务逻辑层和数据层。在表现层，浏览器支持动态的网页和用户接口处理，如身份验证等操作均可以直接在浏览器上处

理。业务逻辑层负责针对具体问题的操作，也可以说是对数据层的操作，对数据业务逻辑处理。系统主要功能和业务逻辑都在业务逻辑层进行处理。数据层则负责提供全局的数据存取模型，以存取各种类型的数据。

第二代的 Internet 克服了第一代 Internet 的弊端，似乎达到了理想的阶段，但是随着用户需求的改变，用户已经不能再满足于只使用 pc 上的软件和只能从 PC 上网。人们希望的是一个更加方便、快捷的网络服务。面对这样的趋势，微软提出了其全新构架——Microsoft.NET。

2000 年 6 月，微软推出了 Microsoft.NET 平台，时任微软首席执行官的鲍尔默这样描述了 Microsoft.NET，他说："Microsoft.NET 代表了一个集合、一个环境、一个可以作为平台支持下一代 Internet 的可编程结构。"

微软的构想是：不再关注单个网站、单个设备与 Internet 相连的互联网环境，而是要构建让所有的计算机群、相关设备和服务商协同工作的网络计算环境。简而言之，互联网提供的服务，要能够完成更高程度的自动化处理。

未来的互联网，应该以一个整体服务的形式展现在最终用户面前，用户只需要知道自己想要什么，而不需要一步步地在网上搜索、操作来达到自己的目的。Microsoft.NET 谋求的是一种理想的互联网环境。而要搭建这样一种互联网环境，首先需要解决的问题是针对现有 Internet 的缺陷，来设计和创造下一代 Internet 结构。这种结构不是物理网络层次上的拓扑结构，而是面向软件和应用层次的、一种有别于浏览器只能静态浏览的可编程 Internet 软件结构。因此 Microsoft.NET 把自己定位为：作为平台支持下一代 Intemet 的可编程结构。

第三节　.NET Framework 应用程序编译与运行的研究分析

C# 和 VB.NET 的代码首先被编译为微软中间语言（Microsoft Intermediate vLanguage，MSIL），并存储在本地。当需要运行这些托管代码时，CLR 又对 MSIL 进行第二次编译（JIT 编译），将 MSIL 代码转换成本机代码（本机代码是针对当前 CPU 的可执行二进制代码）。

对于 .NET Framework 应用程序，安装在用户计算机上的程序并不是针对特定的 CPU 二进制代码，而是此应用程序的 MSIL，一种类似于以前隐藏在编译器内部的中间语言的代码。这么做可以实现可移植性。.NET Framework 的核心并不需要在特定的操作系统和 CPU 上运行，也就是说它可以在非 Windows 操作系统和非 Intel 兼容处理器的系统上运行。

为了实现跨语言应用等目标，微软采取了两项措施：一是发展了一种叫作MSIL 的中间语言，它相当于是一种为虚拟机配套的汇编语言，该语言也叫作 IL（Intermediate Language）；二是提出了一个编译器的制作规范，这个规范叫作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。

每种 .NET Framework 支持的语言编译器都可以生成自描述的、托管的中间语言（Microsoft Intermediate Language，MSIL）代码。所有托管代码都通过使用公共语言运行库运行，公共语言运行库提供了跨语言集成、自动内存管理、跨语言异常处理、增强的安全性的编程模型。

一、编译器选项

下面编译和运行最简单的 C# 程序，这是一个简单的控制台应用程序，它由把某条消息写到屏幕上的一个类组成。在文本编辑器里输入下面的代码，并保存为后缀名为 .cs 的文件（如 HelloDemoCS.cs）。

```
using System;
class MainAPP
{
    public static void Main()
    {
        Console.WriteLine( “hello world using c#!” );
        Console.WriteLine( “hello world using c#!” );
        Console.WriteLine( “hello world using c#!” );
        Console.WriteLine( “hello world using c#!” );
        Console.WriteLine( “hello world using c#!” );
        Console.WriteLine( “hello world using c#!” );
    }
}
```

接下来编译这个程序。.NET Framework 包含一个 C# 的命令编译器，此编译器名为 csc.exe。在 C# 中，通过使用 /? 开关选项可以获得完整的命令行选项列表，如：esc/?

命令行选项包括：/t 开关，指定编译目标；/r 开关，指定引用程序集；/out 开关，指定输出文件的名称。

1. 直接从命令行窗口中编译。

要编译一个 HelloDemoCS.cs 应用程序的源代码，输入：csc HelloDemoCS.cs

这个语法调用 C# 编译器。在这个例子中，只需要指定要编译的文件名称。编译器将产生出可执行文件 HelloDemoCS.exe。

2. 用 /target 或 /t 指定编译目标。

可使用 /target 或 /t 开关选项指定目标文件类型，例如：csc /t:exe HelloDemoCS.es

3. 用 /reference 或 /r 来引用程序集。

当引用其他程序集时应该使用 /reference 或 /r 编译开关选项，例如：esc /t:exe/rrassembl.dll HelloDemoVB.vb

托管代码是面向公共语言运行时的代码。创建托管代码需要如下步骤：

1. 选择编译器

为了获得公共语言运行库提供的优点，必须使用一个或多个针对运行库的语言编译器。

2. 将代码编译为 MSIL

编译器将源代码编译为 MSIL 并生成所需要的元数据。在执行时，实时（JIT）编译器将 MSIL 编译为本机代码。在此编译过程中，代码必须通过验证过程，该过程检查 MSIL 和元数据，以查看是否可以将代码确定为类型安全。托管代码在公共语言运行库提供的虚拟环境中执行，从而可以获得公共语言运行库提供的各种服务。

二、托管执行的过程

在 .NET Framework 中，公共语言运行库为托管环境提供了基础结构。

（一）将源代码编译为托管模块

可以用自己喜欢的语言来编写代码，前提是使用的该语言编译器能够编译面向 CLR 的代码。微软已经创建了几种面向 CLR 的语言编译器：C#、VisualBasic、JScript、J#（一个 Java 语言编译器），以及一个中间语言汇编器。除此之外，其他一些公司也在创建面向 CLR 的编译器，如 COBOL、Fortran 等。在这里可以将编译器看作是一个语法检查器和“正确代码”的分析器。它们对源代码进行检查，确保我们编写的所有东西都有意义，最后输出描述我们意图的指令序列。但是，不管用哪种编译器，最后生成的结果都是一个托管模块。托管模块是一个需要 CLR 才能执行的标准 Windows 可移植可执行文件（PortableExecutable，简称 PE 文件）。托管模块主要包括：编译器将源代码编译为 MSIL，这是一组可以有效地转换为本机代码且独立于 CPU 的指令，可以将其看成是一种“面向对象”的汇编语言，它提供了许多功能强大的 IL 指令，比如 call 指令用于调用一个方法，newobj 指令用于创建一个对

象。在可以执行代码前，必须将 MSIL 转换为 CPU 特定的代码，这通常是由 JIT 编译器完成的，这部分内容我们稍后介绍。由于公共语言运行库为它支持的每种计算机结构都提供了一种或多种 JIT 编译器，因此可以在任何受支持的 CPU 上对同一组 MSIL 进行 JIT 编译和执行。当编译器产生 MSIL 时，它也产生元数据，元数据描述代码中的类型、每种类型的成员的签名、代码引用的成员和运行库在执行时使用的其他数据，公共语言运行库在运行时将会用到元数据。

（二）执行代码

当用户执行托管程序时，操作系统加载器加载公共语言运行库，由 CLR 负责执行托管模块的 MSIL 代码。

代码运行时，CLR 的类加载器从外部存储器中将模块加载到内存中并向元数据咨询该类的信息。代码的执行过程便是调用方法的过程，加载后，CLR 对方法的 MSIL 执行广泛的分析，当一个方法被第一次调用时，CLR 将检测出该方法引用了哪些类型，举例来说，我们在编程时调用了类库中 System.Console 类的 WriteLine 方法来向控制台输出一些信息，那么当程序中 WriteLine 方法被第一次调用时，CLR 会检测出这个方法引用了类库中的 System.Console 类，然后 CLR 会尝试加载 System.Console 类所在的类库程序集文件，当类库程序集文件被加载时，CLR 将自动检查是否有该程序集的预编译版本存在，如果存在，CLR 将加载预编译的代码，不再需要额外的运行时 JIT 编译，若不存在，则被调用方法的 MSIL 代码需要再经过 JIT 编译器动态地编译为本地 CPU 指令代码执行。在进行即时编译的过程中，CLR 同时检查这些 IL 指令是否违反了一些安全规则，这个过程称为验证过程，下面是验证过程可以检验的一些情形：

1. 不能从未初始化的内存中读取数据。
2. 每个方法调用都必须传入正确的参数个数，并且各个参数的类型要正确匹配。
3. 每个方法的返回值都必须被正确地使用等。

如果 IL 代码被确定是“不安全”的，那么 CLR 会停止编译并中断程序的执行。上面即时编译和代码验证的过程仅仅在第一次调用某个方法时发生。CLR 将编译好的本地代码缓存起来，第二次调用时就直接调用缓存中的本地代码，从而避免了再次编译带来的性能损失。这种进行 JIT 编译然后执行代码的过程一直重复到程序中所有代码执行完成时为止。

在执行过程中，托管代码接受自动内存管理、安全性、跨语言调试支持、增强的部署和版本控制支持等服务。