

Microsoft®



.NET Framework 2.0 程序设计

微软公司 著

 人民邮电出版社
POSTS & TELECOM PRESS



.NET Framework 2.0

程序设计

微软公司 著



人民邮电出版社

北京

图书在版编目 (CIP) 数据

.NET Framework 2.0程序设计 / 微软公司著. -- 北京: 人民邮电出版社, 2010.1 (2014.6 重印)
ISBN 978-7-115-22191-9

I. ①N… II. ①微… III. ①计算机网络—程序设计
IV. ①TP393.09

中国版本图书馆CIP数据核字(2010)第005033号

版权声明

本书的著作权归微软公司所有。未经微软公司书面许可, 本书的任何部分不得以任何形式或任何手段复制或传播。著作权人保留所有权利。

内 容 提 要

目前软件开发行业中应用最广泛的平台之一就 Microsoft .NET Framework。 .NET Framework 为各种应用开发提供了丰富的类库资源。开发者使用这些类库, 可以快速地开发应用程序。本课程的目标就在于培养能熟练使用 .NET Framework 类库的程序设计人员。他们可以使用 VB.NET 或者 C#开发基于 .NET Framework 的应用程序。

全书共分为十七章。分别从各个方面向学生介绍了 .NET Framework 的各个部分和技术要点。同时介绍了如何使用 VB.NET 和 C#进行开发和实现。第一章简要介绍了 Microsoft .NET Framework。第二章介绍了公共语言运行库和公共类型系统。第三章介绍了 .NET Framework 中有关托管代码的编译和执行的相关知识。第四章介绍了事件和委托的概念及其用法。从第五章开始, 本书详细介绍了 .NET Framework 2.0 中的各种类库和它们的使用场合。

通过本课程的学习, 学生可以掌握如何使用 .NET Framework 2.0 提供的功能设计应用程序。

.NET Framework 2.0 程序设计

- ◆ 著 微软公司
责任编辑 刘 浩
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京鑫正大印刷有限公司印刷
- ◆ 开本: 787×1092 1/16
印张: 37.5 2010年1月第1版
字数: 914千字 2014年6月北京第2次印刷

ISBN 978-7-115-22191-9

定价: 127.00 元 (附光盘)

读者服务热线: (010) 67132692 印装质量热线: (010) 67129223
反盗版热线: (010) 67171154

编 委 会

编审委员 刘志鹏 林 波 郑祖宪 王 林 田本和

组织策划 林 波 蒋 斌 尹 洪 Jim DiIanni

徐志献 李朝晖 周雨阳 冯 英

技术编审 蒋 斌 虞谷晔 张 充 韩 俊 张 奔

高 娟 李朝晖 张广军 洪国芬

前 言

目前软件开发行业中应用最广泛的平台之一就 Microsoft .NET Framework。 .NET Framework 为各种应用开发提供了丰富的类库资源。开发者使用这些类库，可以快速地开发应用程序。本课程的目标就在于培养能熟练使用 .NET Framework 类库的程序设计人员。他们可以使用 VB.NET 或者 C# 开发基于 .NET Framework 的应用程序。

通过本课程的学习，学生可以掌握如何使用 .NET Framework 2.0 提供的功能设计应用程序：

具体技能包括：

- 能够理解托管执行环境的工作原理。
- 能够进行程序的部署与版本的有效控制。
- 能够使用字符串的各种不同类型的方法及 .NET Framework 的集合和泛型。
- 了解委托，多路委托，事件的概念。
- 了解 .NET 如何进行内存和资源的管理。
- 了解如何使用 GDI+ 进行图形操作。
- 能够操作流，并能使用 Reader 和 Writer 的方法进行对于文本的操作。
- 能够实现序列化。
- 了解线程和异步编程。
- 了解托管代码与非托管代码之间的互操作及在 COM 对象中调用 .NET 对象的方法。
- 了解属性的概念并能定义自定义属性。
- 了解加密的原理并使用相应的类进行加密。
- 了解创建全球化应用程序的过程。
- 了解代码访问安全性的实现原理。
- 能够开发服务应用程序，并熟练使用邮件。

本书适用对象

本书主要面向面向希望熟练使用 .NET Framework 类库的读者。

在学习本课程以前，需具备下列先修课程中的应用操作技能：

- 《Visual C# 2005 程序设计语言》或《Visual Basic 2005 程序设计语言》；
- 《数据库基础》
- 《数据访问技术——ADO.NET 2.0》
- 《Web 应用开发——ASP.NET 2.0》或《Windows 应用开发》

本书结构

本教材的设计策略是遵循由浅入深，循序渐进的策略进行设计的。了解本书的设计策略，对于使用本书教学会有所帮助。

全书共分为十章，第一章介绍了数据库的基础知识和 SQL 语言。这章是现代数据库技术的基础知识，为后续章节的学习打下基础。

第二章从数据访问模型入手，介绍了数据访问模型的基本概念以及主流数据访问模型的发展简史，并在此基础上重点阐述了 ADO.NET 2.0 对象模型及其特征和新特性。

第三章和第四章主要围绕着 ADO.NET 在连接环境中如何连接数据源，并从数据源中获取数据，介绍了相应的对象，方法和操作。

第五章到第七章主要围绕着 ADO.NET 中非常重要的断开连接的环境，介绍了 DataSet 和类型化 DataSet 的概念，并阐述了如何通过 DataAdapter 实现在连接模型和断开连接的模型之间的桥梁作用。

第八章介绍了如何对 DataSet 对象中的数据进行排序、搜索和筛选的各种对象和工具，同时引入了 DataView 对象来使得排序和筛选更加方便。

第九章则阐述了 ADO.NET 中的事务这个概念。第十章主要以 ADO.NET 与 XML 之间的关系为主，介绍了 ADO.NET 操作 XML 的方法。学完这章可以掌握使用 XML 文档和 ADO.NET 之间移动数据的方法，在实际运用中能够同时使用两种技术（ADO.NET 和 XML）编写数据库应用程序。

教学参考资料

获得微软授权使用该教材进行教学的教师，除了可以获得上述材料外，还可以从微软公司获得如下教学资料。

教参	章节教参与教学大纲
实验答案	实验的具体操作步骤
虚拟机	配置好的虚拟机实验环境，用于进行实验和课堂演示
实验室安装指南	包括在实验室部署虚拟机的指南
PowerPoint 和多媒体	用于进行课堂演示
案例	本书随附实用程序训练案例

目 录

第 1 章 .NET Framework 2.0 简介	1	3.2.1 Microsoft 中间语言	25
1.1 .NET Framework 概述	1	3.2.2 元数据	26
1.2 .NET Framework 的结构	3	3.2.3 属性	27
1.2.1 公共语言运行库	3	3.3 组织托管代码: 程序集	27
1.2.2 .NET Framework 类库	3	3.3.1 程序集的元数据: 清单	28
1.3 常见的.NET Framework 应用 程序	5	3.3.2 程序集分类	29
1.3.1 ASP .NET Web 应用程序	5	3.4 执行托管代码	30
1.3.2 Windows 窗体应用程序	6	3.4.1 加载程序集	30
1.3.3 ADO .NET	6	3.4.2 编译 MSIL	31
1.3.4 分布式应用程序	7	3.4.3 垃圾回收	33
1.4 .NET Framework 的运行环境	8	3.4.4 终结器	34
1.5 小结	9	3.4.5 应用程序域	35
1.6 习题	9	3.5 小结	36
第 2 章 公共语言运行库和类型	10	3.6 实验	36
2.1 构建托管代码: 通用类型系统	10	3.7 习题	36
2.2 通用类型系统	10	第 4 章 委托和事件	37
2.2.1 值类型和引用类型	11	4.1 委托	37
2.2.2 值类型和引用类型之间的 相互转换: 装箱和拆箱	12	4.1.1 使用 Delegate 类间接调用 应用程序方法	39
2.2.3 类型转换	14	4.1.2 使用新的 C# 2.0 技术增强 委托行为	45
2.3 公共语言规范	15	4.2 事件	47
2.4 特殊系统类型	16	4.2.1 使用事件的好处	48
2.4.1 泛型	16	4.2.2 事件的工作方式	48
2.4.2 Nullable 类型	18	4.2.3 使用 Event 语句创建 事件	48
2.5 小结	21	4.2.4 EventHandler 委托的 实现	50
2.6 实验	21	4.2.5 自定义事件参数类	51
2.7 习题	21	4.2.6 事件和委托的关系	53
第 3 章 托管代码的编译和执行	24	4.3 小结	53
3.1 编译和执行概述	24	4.4 实验	54
3.2 编译托管代码	25	4.5 习题	54

第 5 章 读取和写入文件	57	5.6.2 使用正则表达式类分析 文本模式	94
5.1 管理文件系统	57	5.7 小结	99
5.1.1 使用 Path 类访问文件 路径	57	5.8 实验	99
5.1.2 使用 File 和 FileInfo 类 访问文件	59	5.9 习题	100
5.1.3 使用 Directory 和 DirectoryInfo 类访问目录	61	第 6 章 集合和泛型	102
5.1.4 使用 DriveInfo 类访问 驱动器	62	6.1 集合和集合接口	102
5.1.5 FileSystemWatcher 类	64	6.1.1 集合	102
5.2 使用字节流	66	6.1.2 集合接口	102
5.2.1 使用 Stream 类管理 字节流	67	6.2 使用主要集合类型	104
5.2.2 使用 FileStream 类管理 文件数据	68	6.2.1 通过迭代器循环访问类的 成员	105
5.2.3 使用 MemoryStream 类 管理内存数据	68	6.2.2 根据键/值对和比较器访问 引用类型	109
5.2.4 使用 BufferedStream 类 提高流性能	69	6.3 使用泛型集合	113
5.3 压缩和保护流信息	70	6.3.1 使用泛型 List 类型创建 类型安全的集合	113
5.3.1 压缩和解压缩	70	6.3.2 使用泛型 Stack 集合和 Queue 集合	114
5.3.2 独立存储	76	6.4 使用专用集合	116
5.3.3 使用独立存储类保护流 信息	78	6.4.1 StringCollection 类	117
5.4 管理应用程序数据	81	6.4.2 StringDictionary 类	118
5.4.1 文本、流、字符串和 二进制数据	81	6.4.3 StringEnumerator 类	118
5.4.2 管理文本数据和字符串	82	6.4.4 CollectionUtil 类	119
5.4.3 管理字符串	83	6.4.5 ListDictionary 类	120
5.4.4 使用 BinaryReader 和 Binary Writer 类管理二进制数据	85	6.4.6 HybridDictionary 类	121
5.5 高效操作字符串	87	6.4.7 OrderedDictionary 类	122
5.5.1 字符串处理	87	6.4.8 NameValueCollection 类	123
5.5.2 使用 StringBuilder 类高效 操作字符串	88	6.4.9 使用专用位结构在内存中 高效地存储数据	124
5.6 使用正则表达式	91	6.5 使用集合基类	127
5.6.1 正则表达式的概念	92	6.5.1 使用集合基类创建自定义 集合	127
		6.5.2 CollectionBase 类	127
		6.5.3 ReadOnlyCollection Base 类	130
		6.5.4 DictionaryBase 类	131
		6.6 小结	134

6.7 实验	135	8.1.6 使用 Font 类在绘制表面 写入文本	219
6.8 习题	135	8.2 操作图形对象的形状和大小	224
第 7 章 数据的序列化	137	8.2.1 使用 Rectangle 类型绘制 图形形状	225
7.1 生成序列化的二进制格式和 SOAP 格式	137	8.2.2 使用 Point 和 Size 类型 指定对象大小	227
7.1.1 BinaryFormatter 类的 成员	139	8.3 使用图像、位图和图标	228
7.1.2 SoapFormatter 类的成员	140	8.3.1 使用 Image 和 Bitmap 类将 图像添加到绘制区域	228
7.2 生成序列化的 XML 格式	141	8.3.2 使用 Icon 类将图标插入 绘制表面	231
7.2.1 使用 XmlSerializer 类将对 象序列化为 XML 格式	142	8.4 小结	233
7.2.2 使用 XML 序列化属性 控制生成的 XML	143	8.5 实验	233
7.3 创建自定义序列化类	150	8.6 习题	233
7.3.1 使用序列化类型收集序列 化信息	151	第 9 章 在 .NET Framework 2.0 中实现 加密	235
7.3.2 使用序列化接口创建 自定义类	154	9.1 加密数据	235
7.3.3 使用格式化程序类将数据 转换为序列化格式	168	9.1.1 数据加密和数据解密	235
7.3.4 使用事件处理程序属性 处理序列化事件	181	9.1.2 使用对称算法类执行 对称加密	236
7.3.5 使用 ObjectManager 类 管理反序列化的对象	196	9.1.3 使用非对称类执行 非对称加密	244
7.4 小结	203	9.1.4 使用 SslStream 类保护 TCP/ IP 通信的安全	248
7.5 实验	203	9.2 计算数据的哈希值	261
7.6 习题	203	9.3 加密行为的扩展	273
第 8 章 GDI+	205	9.3.1 使用加密类管理配置 信息	273
8.1 使用图形、画笔、钢笔、颜色和 字体	205	9.3.2 使用 DPAPI 类保护文件和 内存中的数据	276
8.1.1 图形设备接口	206	9.3.3 使用 CspParameters 类自 定义 CSP 对象的行为	279
8.1.2 使用 Graphics 类创建 绘制表面	206	9.3.4 使用 CryptoAPITransform 类修改加密信息	281
8.1.3 使用 Pen 类绘制直线	210	9.3.5 为加密函数生成随机数	282
8.1.4 使用 Brush 类为图形对象 填充颜色	212	9.4 小结	283
8.1.5 使用 Color 类为图形对象 应用颜色	215	9.5 实验	283

9.6 习题	283	10.6 习题	319
第 10 章 COM 组件与 .NET Framework 程序集之间的交互操作	286	第 11 章 使用类型元数据	321
10.1 使用 Interop 服务访问 COM 组件	286	11.1 通过预定义的 Assembly 类使用类型元数据	321
10.1.1 通过导入类型库创建 Interop 程序集	286	11.1.1 反射	321
10.1.2 在托管代码中使用 COM 的数据类型与 COM 组件进行交互操作	294	11.1.2 使用 Assembly 类访问类型元数据	322
10.1.3 编译和部署 Interop 应用程序的方法	294	11.1.3 使用 MemberInfo 类研究类型元数据	326
10.2 使用 Interop 服务向 COM 组件公开程序集	295	11.1.4 使用 MethodBody 类检查方法的内容	335
10.2.1 与 COM 组件进行交互操作的 .NET Framework 类型	295	11.1.5 使用程序集属性向元数据添加自定义信息	337
10.2.2 应用属性控制 COM 互操作性的类型转换	298	11.2 通过自定义类动态使用程序集	339
10.2.3 打包和部署程序集以实现与 COM 组件的互操作	303	11.2.1 使用生成器类动态创建程序集	339
10.3 使用平台调用服务访问 COM 组件	306	11.2.2 绑定	347
10.3.1 创建保存 Win32 API 函数的 .NET Framework 类	306	11.2.3 使用绑定类型控制成员绑定	348
10.3.2 在托管代码中创建原型	307	11.3 小结	353
10.3.3 在托管代码中调用 COM DLL 函数	310	11.4 实验	353
10.3.4 将 Exception 类映射到 HRESULT	312	11.5 习题	353
10.3.5 平台调用封送数据的方法	314	第 12 章 创建多线程应用程序和应用程序域	355
10.3.6 使用 Marshal 和 MarshalAs Attribute 类封送数据	315	12.1 管理同步环境中的线程	355
10.4 小结	319	12.1.1 线程	356
10.5 实验	319	12.1.2 使用 Thread 类管理线程	356
		12.1.3 使用 ThreadPool 类管理线程池	362
		12.2 管理异步环境中的线程	364
		12.2.1 异步编程	364
		12.2.2 使用异步类管理回调方法	365
		12.2.3 通过异步调用迁移线程的执行上下文	368

12.2.4	使用 Synchronization Context 类管理异步 环境	371	13.3.1	权限	410
12.3	应用程序域的工作原理	374	13.3.2	使用 CodeAccessPermission 类实现权限类型	411
12.3.1	应用程序域	374	13.3.3	配置代码访问安全性 权限	412
12.3.2	使用 AppDomainSetup 类配置应用程序域	375	13.3.4	使用权限集类管理 权限组	420
12.3.3	使用 AppDomain 类创建 应用程序域	376	13.4	管理访问控制	423
12.3.4	从应用程序域检索设置 信息	378	13.4.1	访问控制基类的角色 ..	423
12.3.5	将程序集加载到应用 程序域	379	13.4.2	使用访问控制列表类 管理用户对资源的 访问	424
12.3.6	使用 AppDomain 类卸载 应用程序域	380	13.4.3	使用资源安全类保护 资源	428
12.4	小结	381	13.5	管理用户标识信息	436
12.5	实验	382	13.5.1	IIdentity 接口和 IPrincipal 接口的角色	436
12.6	习题	382	13.5.2	使用 GenericIdentity 类 管理用户标识	437
第 13 章	代码访问安全性	385	13.5.3	使用 Windows 标识类确定 Windows 用户身份	439
13.1	实现代码访问安全性	385	13.5.4	使用 IdentityReference 类 收集用户标识信息	442
13.1.1	代码访问安全性	385	13.5.5	使用 WindowsImpersonation Context 类临时模拟 用户	445
13.1.2	使用 .NET Framework 配置工具配置安全性	387	13.6	小结	448
13.1.3	使用证据类型确定程序 集权限	389	13.7	实验	448
13.2	管理安全性策略	392	13.8	习题	448
13.2.1	安全性策略	392	第 14 章	监视和调试应用程序	451
13.2.2	使用 SecurityManager 类 配置安全性策略	393	14.1	管理事件日志	451
13.2.3	使用策略类管理安全性 策略	394	14.2	应用程序进程的工作原理	453
13.2.4	使用代码组类配置 代码组	396	14.2.1	检索所有正在运行的 进程的列表	453
13.2.5	使用条件类管理代码 组成员	399	14.2.2	检索关于当前进程的 信息	455
13.2.6	使用安全性策略接口创 建自定义安全性策略	403	14.2.3	检索进程所使用的所有 模块的列表	456
13.3	管理权限	410			

14.2.4	启动和停止应用程序 进程	459	第 15 章	使用服务应用程序和电子 邮件消息	493
14.3	管理应用程序的性能	461	15.1	使用 Windows 服务应用程序	493
14.3.1	使用性能监视器监视 应用程序的性能	461	15.1.1	服务应用程序	493
14.3.2	使用性能计数器类 自定义性能信息	461	15.1.2	使用 ServiceBase 类创建 Windows 服务	494
14.4	调试应用程序	464	15.1.3	使用 ServiceInstaller 类 安装服务应用程序	502
14.4.1	使用可视化调试器检查 应用程序错误	465	15.2	使用 ServiceController 类 控制 Windows 服务	505
14.4.2	使用 Debugger 类以编程 方式进行调试	465	15.3	使用电子邮件消息	507
14.4.3	使用 Debug 类以编程方式 进行调试	466	15.3.1	使用邮件类创建电子 邮件消息	507
14.4.4	使用 Debugger 属性配置 用户定义的类型	469	15.3.2	使用 MailAttachment 类 向电子邮件消息添加 资源	510
14.4.5	StackFrame 类	470	15.3.3	使用 SmtpClient 类发送 电子邮件消息	515
14.4.6	StackTrace 类	471	15.3.4	使用 SMTP 异常类处理 电子邮件异常	517
14.5	跟踪应用程序	474	15.3.5	使用 SendCompleteEvent Handler 处理电子邮件 完成事件	518
14.5.1	使用 Trace 类以编程方式 跟踪应用程序	474	15.4	小结	520
14.5.2	使用 TraceSource 类确定 跟踪源	476	15.5	实验	521
14.5.3	使用 TraceSwitch 类配置 跟踪输出	477	15.6	习题	521
14.5.4	使用 TraceListener 类定 向跟踪输出	478	第 16 章	创建全球化应用程序	523
14.5.5	使用 CorrelationManager 类分类跟踪信息	481	16.1	使用全球化类处理文化环境 信息	523
14.6	嵌入管理信息和事件	483	16.1.1	全球化	524
14.6.1	管理类	484	16.1.2	使用 CultureInfo 类访问 文化环境	525
14.6.2	为应用程序预订管理 事件	485	16.1.3	使用 RegionInfo 类访问 区域信息	529
14.6.3	通过 WMI 检索系统资源 信息	487	16.1.4	使用 DateTimeFormatInfo 类格式化某种文化环境 中的日期/时间值	532
14.7	小结	490			
14.8	实验	491			
14.9	习题	491			

16.1.5 使用 NumberFormatInfo 类 格式化数字值	536	16.5 小结	555
16.1.6 使用 CompareInfo 类比较 文化环境信息	539	16.6 实验	555
16.2 创建自定义的文化环境	542	16.7 习题	555
16.3 使用主要编码类	545	第 17 章 配置和安装程序集	557
16.3.1 字符编码	545	17.1 安装程序集	557
16.3.2 使用 Encoding 和 EncodingInfo 类对字符 进行编码	546	17.1.1 创建程序集	558
16.3.3 使用 ASCII 和 Unicode 标准对字符进行编码	548	17.1.2 使用全局程序集缓存 共享程序集	561
16.4 使用高级编码类	551	17.1.3 安装程序集的方法	563
16.4.1 使用 Encoder 和 Encoder Fallback 类处理故障 事件	552	17.2 配置程序集	570
16.4.2 使用 Decoder 和 Decoder Fallback 类处理故障 事件	552	17.2.1 配置文件介绍	570
		17.2.2 访问和管理配置文件	571
		17.2.3 配置文件高级应用	575
		17.3 小结	578
		17.4 实验	578
		17.5 习题	578
		词汇表	580

第 1 章 .NET Framework 2.0 简介

作为一名 C# 或者 VB.NET 程序员，你可能经常会在程序中引用别的类库文件，例如编写 Windows 窗体程序时，无论使用 C# 还是 VB.NET，都需要添加对 System.Windows.Forms.dll 文件的引用。不仅如此，你还会发现无论是 VB.NET 或者 C#，它们虽然语法和结构不同，但是它们所调用的系统类库是一样的，例如在 C# 中有 System.Collections.ArrayList 类，在 VB.NET 中也有 System.Collections.ArrayList 类，而且在这两种语言中这个类提供的功能完全相同。诸如以上种种事实让我们隐隐约约感觉到了一件事情，那就是在 VB.NET 和 C# 之下、操作系统之上有一个共同的运行平台，为这些语言提供了统一的类库和功能调用。作为一名开发人员，你一直在使用这个平台，但是这个平台是什么呢？那就是 .NET Framework。本章将带你了解 .NET Framework 的基本结构、基本功能和工作方式。

在本章中可以学习以下内容。

- 了解什么是 .NET Framework。
- 了解 .NET Framework 的构成。
- 知道常见的 .NET Framework 应用程序种类。

1.1 .NET Framework 概述

.NET Framework 是用于代码编译和执行的集成托管环境，换句话说，它管理着应用程序运行的方方面面，包括程序首次运行的编译、为程序分配内存以存储数据和指令、对应用程序授予或拒绝相应的权限、启动并管理应用程序执行，并且管理剩余内存的再分配。由于所有的 .NET 应用程序都在 .NET Framework 上执行，所以开发人员只需考虑如何与 .NET Framework 打交道，而不必关心 .NET Framework 底层的实现。 .NET Framework 由公共语言运行库（Common Language Runtime, CLR）和 .NET Framework 类库两个主要组件组成。

公共语言运行库可视为管理代码执行的环境。它介于操作系统和应用程序之间，提供了代码编译、内存分配、线程管理以及垃圾回收之类的核心服务。它还强制实施了严格的类型安全检查，并通过强制实施代码访问安全来确保代码在安全的环境中执行。

.NET Framework 提供了一整套很有用且可重用的类型，这套类型是面向对象的且完全可扩展的，它简化了 .NET Framework 应用程序的开发。

.NET 应用程序运行在公共语言运行库之上，并且它还可以使用类库中它所需要的任何部分，如图 1-1 所示。

仅使用 .NET Framework 公共语言运行库提供的功能的代码称为托管代码。如图 1-2 所示，可以完全依赖于公共语言运行库以及 .NET Framework 类库的相关部分，仅使用托管代码构建应用程序。当然，也可以通过组合托管代码和非托管代码来构建应用程序，这时两种代码

根据需要进行交互。第二种选择（如图 1-2 的右半部分所示）对于现有应用程序尤其重要。现在创建的大部分新的 Windows 应用程序都是用托管代码构建的，但是使用托管代码扩展 .NET 之前就已存在的应用程序同样很有用处。例如，企业可以重用 .NET 问世之前开发的程序，使用它提供的功能（非托管代码），同时使用公共语言运行库提供的功能（托管代码）。当然，也可以像 .NET Framework 问世以前一样，完全使用非托管代码创建 Windows 应用程序，.NET Framework 并不是非用不可。

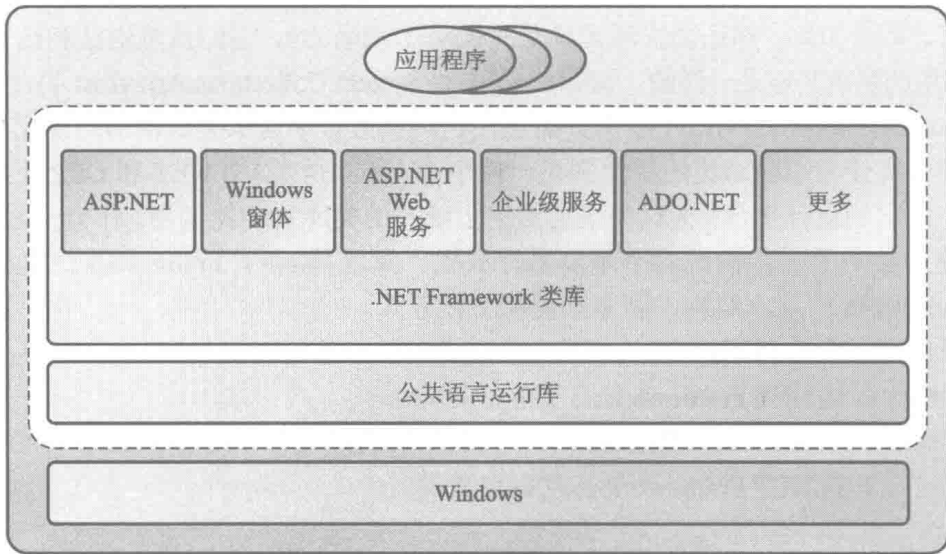


图 1-1 .NET Framework 结构

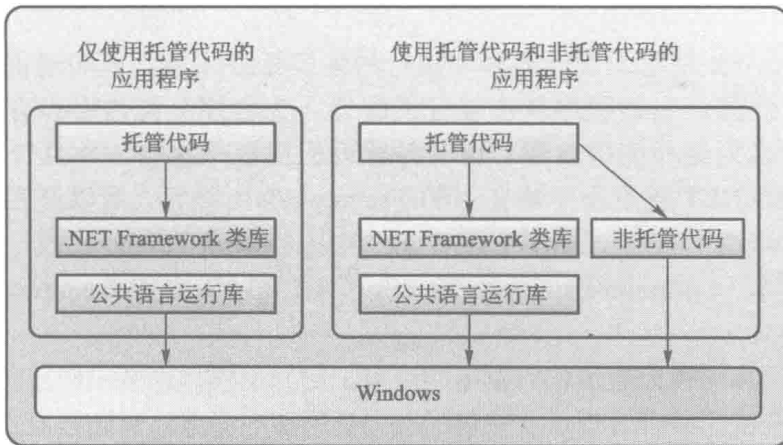


图 1-2 托管代码和非托管代码

托管代码通常是面向对象的，因此它所创建和使用的对象称为托管对象。一个托管对象可使用和继承另一个托管对象，即使两个对象是用不同的语言编写的。这是 .NET Framework 类库成为开发代码的有效基础的关键所在：即以任何基于公共语言运行库的语言编写的对象都可继承和使用该类库的代码。例如，用户完全可以使用 C# 编写一个类，在 VB.NET 中继承这个类或者直接使用这个类。由于公共语言运行库是 .NET Framework 中所有其他内容的基础，因此我们将从理解公共语言运行库入手来深入了解 .NET Framework。

1.2 .NET Framework 的结构

在图 1-1 中，使用 .NET Framework 编写的每个应用程序都依赖于公共语言运行库。公共语言运行库提供了一组通用数据类型，这些类型充当 C#、VB.NET 以及任何其他针对 .NET Framework 的语言的基础。例如，VB.NET 中提供了 Integer 类型，C# 中提供了 System.Int32 类型，它们都与公共语言运行库中的 System.Int32 类型相对应。由于无论开发人员选择何种语言这一基础都是相同的，因此开发人员将看到更为一致的环境。

1.2.1 公共语言运行库

请先思考一下我们以前所学的编程语言，如 VB.NET 和 C#，它们通常是如何定义的。每种语言一般都有自己独特的语法、自己的控制结构集、独特的数据类型集和自己的类继承概念等多种内容。例如，VB.NET 使用换行结束当前语句，C# 中则使用“;”来结束当前语句。目前对于现代编程语言所应提供的多数功能，人们已达成了广泛的共识。虽然开发人员对于编程语言的语法不能达成一致意见，有的人喜欢花括号，而有的人则不喜欢，但是对于语言应提供的语义仍然存在着广泛的共识。既然如此，为什么不对于这些语义定义一种标准实现，然后允许使用不同的语法来表达这些语义呢？

公共语言运行库利用了不同编程语言的相似性，抽象出了通用类型系统（Common Type System, CTS）。通用类型系统构成了 .NET Framework 的公共语言运行库的基础，它定义了 .NET Framework 中的所有数据类型，并提供了面向对象的模型以及各种语言需要遵循的标准。简单来说，无论我们用 C# 还是 VB.NET 进行开发，它们中的所有数据在编译后都转换成了 CTS 中定义的类型，这就意味着 .NET 代码中的全部数据最终都以相同的数据类型存储。但是公共语言运行库不涉及语法。编程语言的表示形式是怎样的？它是包含花括号、冒号还是其他什么符号？这些问题完全由编程语言设计者决定。公共语言运行库本身为编程语言设计者提供了构建工作要基于的统一语义集。

公共语言运行库不是以任何特定编程语言定义的。相反，它的功能主要是从现有流行的编程语言中派生的，如 C++、.NET 之前的 VB 版本以及 Java。现在，Microsoft 提供了多种基于公共语言运行库的语言，包括 VB.NET、C#、经过改进的 C++ 和 JScript.NET 以及后来增加的 Visual J#。第三方也提供了构建于公共语言运行库的语言。

1.2.2 .NET Framework 类库

.NET Framework 类库提供了一整套通用功能的标准代码，这些代码包含了开发人员可用于简化其开发工作的类和其他类型的库。例如，System.Windows.Forms 命名空间包含了构建 Windows 窗体及其所使用控件的所有类。虽然这些类本身都是用 C# 编写的，但是以任何 .NET 语言编写的应用程序都可以使用 .NET Framework 类库中的代码，例如，用 C#、VB.NET、C++ 或 .NET Framework 所支持的任何其他语言编写的代码，都可创建这些类的实例并调用其

方法。如果公共语言运行库中的类是可继承的，这些代码还可以从公共语言运行库中的类继承。而且.NET Framework 类库支持跨语言的继承和调试，比如可以在 C#里面继承一个用 VB.NET 定义的类，或使用一段用 VB.NET 编写的代码，这样在 C#程序里一样可以调试 VB.NET 程序。

.NET Framework 类库的内容组织为命名空间树。命名空间是执行相关功能的类型（如类和接口）的逻辑组织单位。例如，System.Windows.Forms 命名空间包含了构建 Windows 窗体及其所使用控件的所有类。每个命名空间还可以包含其他命名空间。图 1-3 所示为.NET Framework 类库命名空间树的极小一部分。

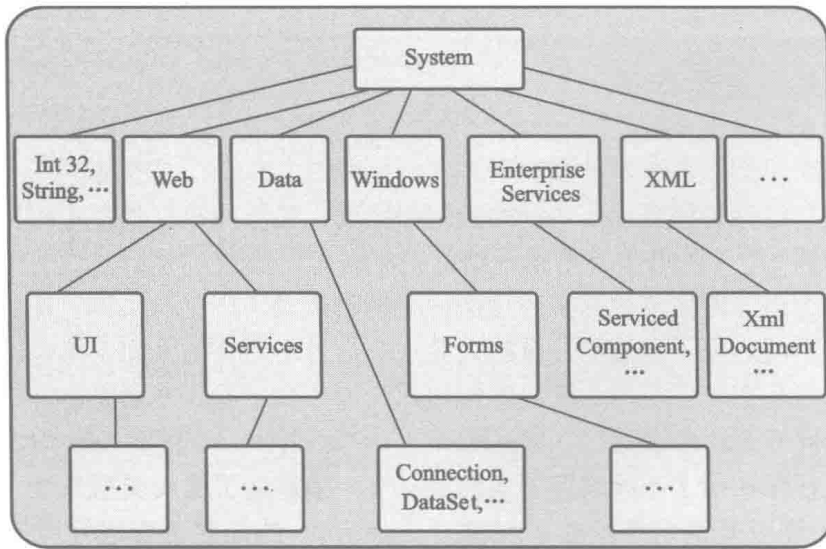


图 1-3 .NET Framework 类库命名空间树

下面简要介绍上图所列出的命名空间。

- **System:** 树的根，此命名空间包含.NET Framework 类库中所有其他命名空间。System 还包含了公共语言运行库以及基于公共语言运行库构建的编程语言所使用的核心数据类型。这些类型包括整型的多种变体、字符串类型以及很多其他类型。
- **System.Web:** 此命名空间包含对于创建 Web 应用程序有用的类型，并且与很多命名空间一样，它拥有下级命名空间。例如，开发人员可使用 System.Web.UI 中的类型来构建 ASP.NET 浏览器应用程序，而 System.Web.Services 中的类型用于构建 ASP.NET Web 服务应用程序。
- **System.Data:** 此命名空间中的类型构成了 ADO.NET。例如，Connection 类用于建立与数据库管理系统（DBMS）的连接，而 DataSet 类的实例用于缓存对该 DBMS 所发出的查询的结果。
- **System.Windows.Forms:** 此命名空间中的类型组成了 Windows 窗体，它们用于构建 Windows 图形用户界面（GUI）。以任何编程语言编写的.NET Framework 应用程序已不再依赖于特定于语言的机制，如 C++中早期的 Microsoft 基础类（MFC），而是使用 System.Windows.Forms 这一通用类型集来构建 Windows 图形用户界面。
- **System.EnterpriseServices:** 此命名空间中的类型提供了企业级应用程序所需要的某