

目 录

第一部分 C#和.NET 平台简介

第 1 章 .NET 之道	2
1.1 了解.NET 之前的世界	2
1.1.1 C/Win32 API 程序员的生活	2
1.1.2 C++/MFC 程序员的生活	2
1.1.3 Visual Basic 6.0 程序员的生活	3
1.1.4 Java/J2EE 程序员的生活	3
1.1.5 COM 程序员的生活	3
1.1.6 Windows DNA 程序员的生活	4
1.2 .NET 解决方案	4
1.3 .NET 平台构造块 (CLR、CTS 和 CLS) 简介	5
1.4 C#的优点	6
1.5 其他支持.NET 的编程语言	6
1.6 .NET 程序集概览	7
1.7 单文件程序集和多文件程序集	8
1.8 CIL 的作用	9
1.8.1 CIL 的好处	11
1.8.2 将 CIL 编译成特定平台的指令	11
1.9 .NET 类型元数据的作用	11
1.10 程序集清单的作用	12
1.11 理解 CTS	12
1.11.1 CTS 类类型	12
1.11.2 CTS 结构类型	13
1.11.3 CTS 接口类型	13
1.11.4 CTS 枚举类型	14
1.11.5 CTS 委托类型	14
1.11.6 CTS 类型成员	14
1.11.7 内建的 CTS 数据类型	14
1.12 理解 CLS	15
1.13 理解 CLR	16
1.14 程序集/命名空间/类型的区别	18
1.14.1 以编程方式访问命名空间	20
1.14.2 引用外部程序集	21

1.15 使用 ildasm.exe	21
1.15.1 查看 CIL 代码	22
1.15.2 查看类型元数据	22
1.15.3 查看程序集元数据	22
1.16 部署.NET 运行库	23
1.17 .NET 的平台无关性	23
1.18 小结	24
第 2 章 构建 C#应用程序	25
2.1 安装.NET Framework 2.0 SDK	25
2.2 C#命令行编译器 (csc.exe)	26
2.2.1 配置 C#命令行编译器	26
2.2.2 配置其他.NET 命令行工具	27
2.3 使用 csc.exe 构建 C#应用程序	27
2.3.1 引用外部程序集	28
2.3.2 使用 csc.exe 编译多个源文件	29
2.3.3 引用多个外部程序集	30
2.4 使用 csc.exe 响应文件	30
2.5 命令行调试器 (cordbg.exe)	31
2.6 使用 TextPad 构建.NET 应用程序	32
2.6.1 启用 C#关键字着色	32
2.6.2 配置*.cs 文件过滤器	33
2.6.3 与 csc.exe 关联	33
2.6.4 将运行命令与菜单项相关联	34
2.6.5 启用 C#代码片段	35
2.7 使用 SharpDevelop 构建.NET 应用 程序	35
2.7.1 SharpDevelop	36
2.7.2 Project Scout 和 Classes Scout	36
2.7.3 Assembly Scout	37
2.7.4 Windows 窗体设计器	38
2.8 使用 Visual C# 2005 Express 构建.NET 应用程序	38
2.9 使用 Visual Studio 2005 构建.NET 应用 程序	39
2.9.1 Visual Studio 2005	39

2.9.2	Solution Explorer 工具	40	3.11.2	out 修饰符	71
2.9.3	Class View 工具	41	3.11.3	ref 修饰符	72
2.9.4	Code Definition 窗口	41	3.11.4	params 修饰符	73
2.9.5	Object Browser 工具	41	3.12	迭代结构	73
2.9.6	集成对代码重构的支持	42	3.12.1	for 循环	73
2.9.7	代码扩展和围绕技术	43	3.12.2	foreach 循环	74
2.9.8	可视化 Class Designer	43	3.12.3	while 和 do/while 循环结构	74
2.9.9	对象测试平台	45	3.13	判断结构与关系/相等运算符	75
2.9.10	集成的帮助系统	46	3.13.1	if/else 语句	75
2.10	其他.NET 开发工具	47	3.13.2	switch 语句	76
2.11	小结	47	3.14	值类型和引用类型	77
第二部分 C#编程语言					
第3章 C#语言基础					
3.1	剖析一个简单的 C#程序	50	3.14.1	值类型、引用类型和赋值运算符	78
3.1.1	Main()方法的其他形式	51	3.14.2	包含引用类型的值类型	79
3.1.2	处理命令行参数	51	3.14.3	按值传递引用类型	81
3.1.3	使用 Visual Studio 2005 指定 命令行参数	52	3.14.4	按引用传递引用类型	82
3.2	有趣的题外话: System.Environment 类	53	3.14.5	值类型和引用类型: 最后的 细节	82
3.3	定义类并创建对象	53	3.15	装箱与拆箱操作	83
3.3.1	构造函数的作用	54	3.15.1	实用的装箱和拆箱示例	84
3.3.2	是内存泄露吗	55	3.15.2	拆箱自定义的值类型	85
3.3.3	定义“应用程序对象”	56	3.16	使用.NET 枚举	86
3.4	System.Console 类	57	3.17	最重要的类: System.Object	89
3.4.1	使用 Console 类进行基本的输 入和输出	57	3.18	重写 System.Object 的一些 默认行为	92
3.4.2	格式化控制台输出	58	3.18.1	重写 System.Object .ToString()	92
3.4.3	.NET 字符串格式化标志	58	3.18.2	重写 System.Object .Equals()	92
3.5	设置成员的可见性	60	3.18.3	重写 System.Object .GetHashCode()	93
3.6	类成员变量的默认值	61	3.18.4	测试重写的成员	94
3.7	成员变量的初始化语法	62	3.18.5	System.Object 的静态成员	95
3.8	定义常量数据	63	3.19	系统数据类型 (和 C#简化符号)	95
3.9	定义只读字段	64	3.19.1	数值数据类型的实验	97
3.10	static 关键字	66	3.19.2	System.Boolean 的成员	97
3.10.1	静态方法	66	3.19.3	System.Char 的成员	98
3.10.2	静态数据	67	3.19.4	从字符串数据中解析数值	98
3.10.3	静态构造函数	68	3.19.5	System.DateTime 和 System.TimeSpan	99
3.10.4	静态类	69	3.20	System.String 数据类型	100
3.11	方法参数修饰符	70	3.20.1	基本的字符串操作	100
3.11.1	默认的参数传递行为	71	3.20.2	转义字符	101

3.20.3 使用 C# 的逐字字符串	102	4.6.1 virtual 和 override 关键字	134
3.21 System.Text.StringBuilder 的作用	102	4.6.2 再谈 sealed 关键字	135
3.22 .NET 数组类型	103	4.6.3 抽象类	136
3.22.1 数组作为参数 (和返回值)	104	4.6.4 强制多态活动: 抽象方法	137
3.22.2 使用多维数组	104	4.6.5 成员隐藏	139
3.22.3 System.Array 基类	106	4.7 C# 的类型转换规则	140
3.23 C# 的可空类型	107	4.7.1 确定 Employee 的类型	141
3.23.1 使用可空类型	107	4.7.2 数值类型转换	142
3.23.2 ?? 运算符	108	4.8 C# 的分部类型	142
3.24 定义自定义命名空间	108	4.9 通过 XML 生成 C# 源代码的文档	144
3.24.1 类型的完全限定名	110	4.9.1 XML 代码注释格式化字符	146
3.24.2 使用别名定义命名空间	111	4.9.2 转换 XML 代码注释	146
3.24.3 创建嵌套的命名空间	112	4.10 小结	146
3.24.4 Visual Studio 2005 中的“默 认命名空间”	113	第 5 章 对象的生命周期	147
3.25 小结	113	5.1 类、对象和引用	147
第 4 章 C# 2.0 面向对象编程	114	5.2 对象生命周期的基础	148
4.1 C# 的类类型	114	5.3 应用程序根的作用	150
4.1.1 方法重载	116	5.4 对象的代	151
4.1.2 使用 C# 的 this 进行自引用	116	5.5 System.GC 类型	152
4.1.3 定义类的公共接口	118	5.6 构建可终结对象	155
4.2 回顾 OOP 的支柱	118	5.6.1 重写 System.Object .Finalize()	155
4.2.1 封装	119	5.6.2 终结过程的细节	157
4.2.2 继承	119	5.7 构建可处置对象	157
4.2.3 多态	120	5.8 构建可终结类型和可处置类型	159
4.3 第一个支柱: C# 的封装支持	121	5.9 小结	161
4.3.1 使用传统的访问方法和 修改方法执行封装	122	第 6 章 结构化异常处理	162
4.3.2 另一种形式的封装: 类属性	122	6.1 错误、bug 与异常	162
4.3.3 C# 属性的内部表示	124	6.2 .NET 异常处理的作用	163
4.3.4 控制属性 get/set 语句的可见 性级别	126	6.2.1 .NET 异常处理的四要素	163
4.3.5 只读和只写属性	126	6.2.2 System.Exception 基类	164
4.3.6 静态属性	126	6.3 最简单的例子	165
4.4 第二个支柱: C# 的继承支持	127	6.3.1 引发普通的异常	166
4.4.1 使用 base 控制基类的创建	128	6.3.2 捕获异常	167
4.4.2 关于多基类	130	6.4 配置异常的状态	168
4.4.3 保护家族的秘密: protected 关 键字	130	6.4.1 TargetSite 属性	168
4.4.4 防止继承: 密封类	130	6.4.2 StackTrace 属性	169
4.5 为包含/委托编程	131	6.4.3 HelpLink 属性	169
4.6 第三个支柱: C# 的多态支持	134	6.4.4 Data 属性	170
		6.5 系统级异常 (System.SystemException)	171

6.6 应用程序级异常 (System.ApplicationException)	171
6.6.1 构建自定义异常, 第一部分	172
6.6.2 构建自定义异常, 第二部分	173
6.6.3 构建自定义异常, 第三部分	173
6.7 处理多个异常	174
6.7.1 通用的 catch 语句	176
6.7.2 再次引发异常	176
6.7.3 内部异常	177
6.8 finally 块	178
6.9 谁在引发什么异常	178
6.10 未处理异常的后果	179
6.11 使用 Visual Studio 2005 调试未处理的异常	180
6.12 小结	180
第 7 章 接口与集合	181
7.1 使用 C# 定义接口	181
7.2 使用 C# 实现接口	182
7.3 接口与抽象基类的对比	183
7.4 在对象级别调用接口成员	184
7.4.1 获取接口引用: as 关键字	184
7.4.2 获取接口引用: is 关键字	185
7.5 接口作为参数	185
7.6 接口作为返回值	187
7.7 接口类型数组	187
7.8 显式接口实现	188
7.9 构建接口层次结构	190
7.10 使用 Visual Studio 2005 实现接口	192
7.11 构建可枚举类型 (IEnumerable 和 IEnumerator)	193
7.12 构建可克隆的对象 (ICloneable)	196
7.13 构建可比较的对象 (IComparable)	199
7.13.1 指定多个排序的顺序 (IComparer)	201
7.13.2 自定义属性、自定义 排序类型	202
7.14 System.Collections 命名空间的接口	203
7.14.1 ICollection 接口的作用	203
7.14.2 IDictionary 接口的作用	204
7.14.3 IDictionaryEnumerator 接口的作用	204
7.14.4 IList 接口的作用	204
7.15 System.Collections 命名空间中的类	205
7.15.1 操作 ArrayList 类型	205
7.15.2 操作 Queue 类型	206
7.15.3 操作 Stack 类型	207
7.16 System.Collections.Specialized 命名空间	208
7.17 小结	208
第 8 章 回调接口、委托与事件	209
8.1 回调接口	209
8.2 .NET 委托类型	212
8.3 使用 C# 定义委托	213
8.4 System.MulticastDelegate 与 System.Delegate 基类	215
8.5 最简单的委托示例	216
8.6 使用委托改造 Car 类型	218
8.7 更复杂的委托示例	222
8.7.1 委托作为参数	223
8.7.2 分析委托代码	225
8.8 委托协变	226
8.9 C# 事件	228
8.9.1 揭开事件的神秘面纱	229
8.9.2 监听传入的事件	230
8.9.3 使用 Visual Studio 2005 简化 事件注册	231
8.9.4 严谨规范的事件	231
8.10 C# 匿名方法	233
8.11 C# 方法组转换	235
8.12 小结	236
第 9 章 高级 C# 类型构造技术	237
9.1 构建自定义索引器	237
9.2 类型索引器的内部表示方式	239
9.3 索引器: 最后的细节	240
9.4 运算符重载	240
9.5 重载二元运算符	241
9.6 重载一元运算符	243
9.7 重载相等于运算符	243
9.8 重载比较运算符	244
9.9 重载运算符的内部表示形式	245
9.10 在不支持重载运算符 的语言中使用重载运算符	246
9.11 运算符重载的最后思考	247

9.12 自定义类型转换.....	247	11.1.3 程序集是可版本化的单元.....	287
9.12.1 回顾: 数值转换.....	247	11.1.4 程序集是自描述的.....	287
9.12.2 回顾: 相关的类类型间的转换.....	247	11.1.5 程序集是可配置的.....	287
9.13 创建自定义转换例程.....	248	11.2 .NET 程序集的格式.....	287
9.14 定义隐式转换例程.....	250	11.2.1 Win32 文件首部.....	288
9.15 自定义转换例程的内部表示.....	251	11.2.2 CLR 文件首部.....	288
9.16 C#的高级关键字.....	252	11.2.3 CIL 代码、类型元数据和程序集清单.....	289
9.16.1 checked 关键字.....	252	11.2.4 可选的程序集资源.....	289
9.16.2 unchecked 关键字.....	254	11.2.5 单文件程序集和多文件程序集.....	289
9.16.3 指针类型.....	254	11.3 构建和使用单文件程序集.....	291
9.16.4 sizeof 关键字.....	259	11.3.1 清单.....	293
9.17 C#预处理指令.....	260	11.3.2 CIL.....	294
9.17.1 指定代码区域.....	260	11.3.3 类型元数据.....	295
9.17.2 条件代码编译.....	261	11.3.4 构建 C#客户端应用程序.....	295
9.18 小结.....	262	11.3.5 构建 Visual Basic .NET 客户端应用程序.....	296
第 10 章 泛型	263	11.3.6 实现跨语言继承.....	297
10.1 再论装箱、拆箱和 System.Object 之间的关系.....	263	11.4 构建和使用多文件程序集.....	298
10.2 装箱/拆箱操作的问题.....	264	11.4.1 ufo.netmodule 文件.....	299
10.2.1 类型安全与强类型集合.....	264	11.4.2 airvehicles.dll 文件.....	299
10.2.2 装箱与强类型集合.....	266	11.4.3 使用多文件程序集.....	299
10.3 System.Collections.Generic 命名空间.....	268	11.5 私有程序集.....	300
10.4 创建泛型方法.....	271	11.5.1 私有程序集的标识.....	300
10.5 创建泛型结构(或类).....	273	11.5.2 探测过程.....	301
10.6 创建自定义泛型集合.....	275	11.5.3 配置私有程序集.....	301
10.6.1 使用 where 约束类型参数.....	276	11.5.4 配置文件和 Visual Studio 2005.....	303
10.6.2 运算符约束的不足.....	278	11.5.5 .NET Framework 2.0 配置工具简介.....	303
10.7 创建泛型基类.....	279	11.6 共享程序集.....	304
10.8 创建泛型接口.....	280	11.6.1 强名称.....	305
10.9 创建泛型委托.....	281	11.6.2 为 CarLibrary.dll 赋予强名称.....	306
10.9.1 在 .NET 1.1 下模拟泛型委托.....	282	11.6.3 使用 Visual Studio 2005 为程序集赋予强名称.....	307
10.9.2 嵌套委托相关简介.....	282	11.6.4 在 GAC 中安装和移除共享程序集.....	307
10.10 小结.....	283	11.6.5 延迟签名的作用.....	308
第三部分 .NET 程序集编程		11.7 使用共享程序集.....	308
第 11 章 .NET 程序集入门	286	11.8 配置共享程序集.....	310
11.1 .NET 程序集的作用.....	286	11.8.1 冻结当前的共享程序集.....	310
11.1.1 程序集促进代码重用.....	286		
11.1.2 程序集确定类型边界.....	287		

11.8.2	构建共享程序集 2.0.0.0 版本	311	12.6.2	调用没有参数的方法	334
11.8.3	动态重定向到共享程序集的特定版本	312	12.6.3	调用有参数的方法	335
11.8.4	再次研究 .NET Framework 2.0 配置工具	313	12.7	特性编程	335
11.9	研究 GAC 的内部结构	313	12.7.1	特性的使用者	336
11.10	发行者策略程序集	315	12.7.2	在 C# 中使用预定义特性	336
11.11	<codeBase> 元素	316	12.7.3	为特性指定构造参数	338
11.12	System.Configuration 命名空间	317	12.7.4	Obsolete 特性	338
11.13	机器配置文件	318	12.7.5	C# 特性简化符号	338
11.14	程序集绑定总体流程图	318	12.8	构建自定义特性	339
11.15	小结	319	12.8.1	应用自定义特性	339
第 12 章	类型反射、晚期绑定和基于特性的编程	320	12.8.2	限制特性使用	340
12.1	类型元数据的必要性	320	12.9	程序集级别（和模块级别）特性	341
12.1.1	查看（部分）EngineState 枚举的元数据	321	12.10	使用早期绑定反射特性	342
12.1.2	查看（部分）Car 类型的元数据	321	12.11	使用晚期绑定反射特性	342
12.1.3	研究 TypeRef	322	12.12	反射、晚期绑定和自定义特性的使用背景	343
12.1.4	记录定义的程序集	323	12.13	构建可扩展的应用程序	344
12.1.5	记录引用的程序集	323	12.13.1	构建 CommonSnappable-Types.dll	344
12.1.6	记录字符串字面量	323	12.13.2	构建 C# 插件	345
12.2	反射	324	12.13.3	构建 Visual Basic.NET 插件	345
12.2.1	System.Type 类	324	12.13.4	构建可扩展的 Windows 窗体应用程序	346
12.2.2	使用 System.Object.GetType() 得到 Type 引用	325	12.14	小结	348
12.2.3	使用 System.Type.GetType() 得到 Type 引用	325	第 13 章	进程、应用程序域、上下文和 CLR 宿主	349
12.2.4	使用 typeof() 得到 Type 引用	326	13.1	回顾传统的 Win32 进程	349
12.3	构建自定义的元数据查看器	326	13.2	.NET 平台下与进程进行交互	351
12.3.1	反射方法	326	13.2.1	列举运行中的进程	352
12.3.2	反射字段和属性	327	13.2.2	研究特定的进程	352
12.3.3	反射实现的接口	327	13.2.3	研究进程的线程集合	353
12.3.4	显示其他信息	327	13.2.4	研究进程中的模块集合	354
12.3.5	实现 Main()	328	13.2.5	以编程方式启动或结束进程	355
12.3.6	反射方法参数和返回值	329	13.3	.NET 应用程序域	356
12.4	动态加载程序集	330	13.3.1	列举进程中的应用程序域	357
12.5	反射共享程序集	332	13.3.2	以编程方式创建新的应用程序域	358
12.6	晚期绑定	333	13.3.3	以编程方式卸载应用程序域	360
12.6.1	System.Activator 类	333	13.4	对象上下文边界	361
			13.4.1	上下文灵活和上下文绑定类型	361

13.4.2	定义上下文绑定对象	362
13.4.3	研究对象的上下文	363
13.5	进程、应用程序域和上下文小结	364
13.6	承载 CLR	364
13.6.1	CLR 的并行执行	365
13.6.2	加载特定的 CLR 版本	366
13.6.3	其他的 CLR 宿主	366
13.7	小结	367
第 14 章	构建多线程应用程序	368
14.1	进程、应用程序域、上下文及线程之间的关系	368
14.2	.NET 委托的简短回顾	369
14.3	委托的异步天性	371
14.3.1	BeginInvoke()和 EndInvoke()方法	371
14.3.2	System.IAsyncResult 接口	372
14.4	异步调用方法	372
14.4.1	同步调用线程	373
14.4.2	AsyncCallback 委托的作用	374
14.4.3	AsyncResult 类的作用	375
14.4.4	传递和接收自定义状态数据	375
14.5	System.Threading 命名空间	376
14.6	System.Threading.Thread 类	377
14.6.1	获得当前线程的统计信息	377
14.6.2	Name 属性	378
14.6.3	Priority 属性	378
14.7	以编程方式创建次线程	379
14.7.1	使用 ThreadStart 委托	379
14.7.2	使用 Parameterized-ThreadStart 委托	381
14.7.3	前台线程和后台线程	382
14.8	并发问题	383
14.8.1	使用 C#的 lock 关键字进行同步	385
14.8.2	使用 System.Threading.Monitor 类型进行同步	386
14.8.3	使用 System.Threading.Interlocked 类型进行同步	387
14.8.4	使用[Synchronization]进行同步	388
14.9	使用 Timer Callback 编程	388
14.10	CLR 线程池	390
14.11	小结	391
第 15 章	CIL 和动态程序集的作用	392
15.1	CIL 编程的本质	392
15.2	研究 CIL 指令、特性和操作码	393
15.2.1	CIL 指令的作用	393
15.2.2	CIL 特性的作用	393
15.2.3	CIL 操作码的作用	393
15.2.4	区别 CIL 操作码和 CIL 助记符	394
15.3	入栈和出栈: CIL 基于栈的本质	394
15.4	正反向工程	395
15.4.1	CIL 代码标签的作用	397
15.4.2	与 CIL 交互: 修改*.il 文件	398
15.4.3	使用 ilasm.exe 编译 CIL 代码	399
15.4.4	使用 SharpDevelop 编译 CIL 代码	400
15.4.5	使用 ILIDE#编译 CIL 代码	400
15.4.6	peverify.exe 的作用	401
15.5	CIL 指令和特性	401
15.5.1	在 CIL 中指定外部引用程序集	401
15.5.2	在 CIL 中定义当前程序集	401
15.5.3	在 CIL 中定义命名空间	402
15.5.4	在 CIL 中定义类类型	402
15.5.5	在 CIL 中定义和实现接口	403
15.5.6	在 CIL 中定义结构	404
15.5.7	在 CIL 中定义枚举	404
15.5.8	编译 CILTypes.il 文件	404
15.6	.NET 基类库、C#和 CIL 数据类型的映射	405
15.7	在 CIL 中定义成员	405
15.7.1	在 CIL 中定义数据字段	405
15.7.2	在 CIL 中定义类型的构造函数	406
15.7.3	在 CIL 中定义属性	406
15.7.4	定义成员参数	407
15.8	剖析 CIL 操作码	407
15.8.1	了解.maxstack 指令	409
15.8.2	在 CIL 中声明局部变量	409
15.8.3	在 CIL 中映射参数到局部变量	410

15.8.4	隐式 this 引用	410	16.7	使用 File 类型	436
15.8.5	在 CIL 中使用循环结构	411	16.8	Stream 抽象类	437
15.9	使用 CIL 构建 .NET 程序集	411	16.9	使用 StreamWriter 和 StreamReader 类型	440
15.9.1	构建 CILCars.dll	412	16.9.1	写文本文件	440
15.9.2	构建 CILCarClient.exe	414	16.9.2	从文本文件读	441
15.10	动态程序集	415	16.9.3	直接创建 StreamWriter /StreamReader 类型	442
15.10.1	System.Reflection.Emit 命名空间	416	16.10	使用 StringWriter 和 StringReader	442
15.10.2	System.Reflection.Emit .ILGenerator 的作用	416	16.11	使用 BinaryWriter 和 BinaryReader	443
15.10.3	产生动态的程序集	417	16.12	以编程方式“观察”文件	445
15.10.4	产生程序集和模块集	419	16.13	实现异步文件 I/O 操作	447
15.10.5	ModuleBuilder 类型的作用	420	16.14	小结	448
15.10.6	产生 HelloClass 类型和 字符串成员变量	420	第 17 章 对象序列化		449
15.10.7	产生构造函数	421	17.1	对象序列化	449
15.10.8	产生 HelloWorld() 方法	422	17.2	为序列化配置对象	451
15.10.9	使用动态产生的程序集	422	17.3	选择序列化格式化程序	452
15.11	System.CodeDom 简单说明	423	17.3.1	IFormatter 和 IRemoting- Formatting 接口	452
15.12	小结	423	17.3.2	在格式化程序中的类型保真	453
第四部分 使用 .NET 库编程					
第 16 章 System.IO 命名空间		426	17.4	使用 BinaryFormatter 序列化对象	453
16.1	研究 System.IO 命名空间	426	17.5	使用 SoapFormatter 序列化对象	455
16.2	Directory (Info) 和 File (Info) 类型	427	17.6	使用 XmlSerializer 序列化对象	456
16.3	使用 DirectoryInfo 类型	428	17.7	持久化对象集合	458
16.3.1	FileAttributes 枚举	429	17.8	自定义序列化过程	459
16.3.2	使用 DirectoryInfo 类型列出 文件	429	17.8.1	深入了解对象序列化	460
16.3.3	使用 DirectoryInfo 类型创建 子目录	430	17.8.2	使用 ISerializable 自定义 序列化	461
16.4	使用 Directory 类型	431	17.8.3	使用特性自定义序列化	462
16.5	使用 DriveInfo 类类型	432	17.9	可序列化对象的版本处理	463
16.6	使用 FileInfo 类	433	17.10	小结	465
16.6.1	FileInfo.Create() 方法	433	第 18 章 .NET 远程处理层		466
16.6.2	FileInfo.Open() 方法	434	18.1	定义 .NET 远程处理	466
16.6.3	FileInfo.OpenRead() 和 FileInfo.OpenWrite() 方法	435	18.2	.NET 远程处理命名空间	466
16.6.4	FileInfo.OpenText() 方法	435	18.3	.NET 远程处理框架	467
16.6.5	FileInfo.CreateText() 和 FileInfo.AppendText() 方法	435	18.3.1	代理和消息	468
			18.3.2	信道	469
			18.3.3	.NET 格式化程序的作用	469
			18.3.4	综合讨论	469
			18.3.5	扩展默认管道的简单介绍	470
			18.4	.NET 远程处理数据交换的术语	470

18.4.1	对象封送方式: MBR 还是 MBV	470	18.20	小结	500
18.4.2	选择 MBR 的激活类型: WKO 还是 CAO	472	第 19 章 使用 System.Windows.Forms 构建更好的窗体		501
18.4.3	WKO 类型的状态配置: 单例还是单一调用	473	19.1	System.Windows.Forms 命名空间概述	501
18.4.4	MBR 对象类型特性小结	473	19.2	使用 Windows 窗体类型	502
18.5	.NET 远程处理项目的基本部署	473	19.2.1	手动创建主窗口	502
18.6	构建第一个分布式应用程序	474	19.2.2	重视分离关注点	503
18.6.1	构建普通程序集	474	19.3	Application 类的作用	504
18.6.2	构建服务器端程序集	475	19.3.1	Application 类的使用	505
18.6.3	建立 SimpleRemoteObject-Client.exe 程序集	476	19.3.2	System.EventHandler 委托	506
18.6.4	测试远程处理应用程序	477	19.4	剖析 Form	506
18.7	ChannelServices 类型	477	19.5	Control 类的功能	507
18.8	RemotingConfiguration 类型	478	19.5.1	Control 类的使用	509
18.9	WKO 类型激活模式	479	19.5.2	响应 MouseMove 事件	509
18.10	把服务器部署成远程机器	480	19.5.3	检测被单击的鼠标键	510
18.11	利用 TCP 通道	480	19.5.4	响应键盘事件	511
18.12	简单谈谈 IpcChannel	481	19.6	Form 类的功能	512
18.13	远程处理配置文件	481	19.7	使用 Visual Studio 2005 构建窗口应用程序	514
18.13.1	构建服务器端*.config 文件	482	19.7.1	启用过时的控件	515
18.13.2	构建客户端*.config 文件	483	19.7.2	研究 Visual Studio 2005 Windows 窗体项目	516
18.14	使用 MBV 对象	484	19.7.3	在设计时处理事件	517
18.14.1	构建普通程序集	484	19.7.4	Program 类	517
18.14.2	构建服务器端程序集	485	19.7.5	被自动引用的程序集	518
18.14.3	构建客户端程序集	486	19.8	MenuStrips 和 ContextMenuStrips 的使用	518
18.15	客户端激活的对象	487	19.8.1	向 MenuStrip 添加 TextBox	520
18.16	CAO/WKO-Singleton 对象基于租约的生存期	489	19.8.2	创建上下文菜单	521
18.16.1	默认的租约行为	489	19.8.3	选择菜单项	523
18.16.2	改变默认租约特性	491	19.9	使用 StatusStrip	524
18.16.3	服务器端租约调整	492	19.9.1	设计菜单系统	524
18.16.4	客户端租约调整	492	19.9.2	设计 StatusStrip	525
18.17	服务器端 (和客户端) 租约主办方机制	493	19.9.3	用 Timer 类型工作	527
18.18	远程对象的其他宿主	494	19.9.4	切换显示	527
18.18.1	使用 Windows 服务承载远程对象	494	19.9.5	显示菜单选择提示符	528
18.18.2	使用 IIS 承载远程对象	497	19.9.6	建立 “Ready” 状态	528
18.19	异步远程处理	498	19.10	使用 ToolStrip 工作	529
			19.11	构建 MDI 运用程序	534
			19.11.1	构建父窗体	534
			19.11.2	构建子窗体	535
			19.11.3	复制子窗体	535

19.12 小结	536	20.14.4 把*.resources 文件绑定到.NET 程序集	571
第 20 章 使用 GDI+绘制图形	537	20.14.5 使用 ResourceWriter	572
20.1 GDI+命名空间概述	537	20.14.6 使用 Visual Studio 2005 生成资源	572
20.2 System.Drawing 命名空间概述	538	20.14.7 通过编程读取资源	574
20.3 System.Drawing 实用类型	538	20.15 小结	574
20.3.1 Point(F)类型	539	第 21 章 Windows 窗体控件编程	575
20.3.2 Rectangle(F)类型	539	21.1 Windows 窗体控件	575
20.3.3 Region 类	540	21.2 手动给窗体添加控件	575
20.4 Graphics 类	541	21.3 使用 Visual Studio 2005 给窗体添加控件	578
20.5 Paint 会话	542	21.4 基本控件的使用	578
20.5.1 使窗体的客户区域失效	543	21.4.1 Label 的作用	579
20.5.2 在 Paint 事件处理程序外获取 Graphics 对象	544	21.4.2 TextBox 的作用	580
20.5.3 关于 Graphics 对象的释放	545	21.4.3 MaskedTextBox 的作用	581
20.6 GDI+坐标系统	546	21.4.4 Button 的作用	583
20.6.1 默认度量单位	546	21.4.5 CheckBox、RadioButton 和 GroupBox 的作用	585
20.6.2 指定另一种度量单位	547	21.4.6 CheckedListBox 的作用	587
20.6.3 指定另一个原点	548	21.4.7 ListBox 的作用	588
20.7 定义颜色值	549	21.4.8 ComboBox 的作用	589
20.8 操作字体	550	21.5 配置选项卡的次序	589
20.8.1 使用字体族	551	21.6 设置窗体的默认输入按钮	590
20.8.2 使用字体名和字体大小	552	21.7 更多奇特的控件	590
20.8.3 枚举安装的字体	553	21.7.1 MonthCalendar 控件的作用	591
20.8.4 FontDialog 类	555	21.7.2 ToolTip 控件的作用	592
20.9 System.Drawing.Drawing2D 命名空间概述	556	21.7.3 TabControl 控件的作用	593
20.10 使用 Pen	556	21.7.4 TrackBar 的作用	593
20.11 使用 Brush	558	21.7.5 Panel 的作用	595
20.11.1 使用 HatchBrush	559	21.7.6 UpDown 控件的作用	596
20.11.2 使用 TextureBrush	560	21.7.7 ErrorProvider 的作用	598
20.11.3 使用 LinearGradientBrushes	561	21.7.8 TreeView 的作用	599
20.12 呈现图像	562	21.7.9 WebBrowser 的作用	603
20.13 PictureBox 控件的拖动和单击测试	563	21.8 创建自定义 Windows 窗体控件	604
20.13.1 呈现图像的单击测试	565	21.8.1 创建图像	605
20.13.2 非矩形图像的单击测试	567	21.8.2 构建设计时 UI	605
20.14 .NET 资源格式	569	21.8.3 实现核心的 CarControl	605
20.14.1 System.Resources 命名空间	570	21.8.4 定义自定义事件	607
20.14.2 以编程方式创建*.resx 文件	570	21.8.5 定义自定义属性	607
20.14.3 构建*.resources 文件	571	21.8.6 控制动画	608
		21.8.7 显示昵称	608

21.9	测试 CarControl 类型	609	22.10.2	使用 .NET 2.0 的 ConnectionStringBuilder	637
21.10	创建自定义 CarControl 窗体宿主	609	22.10.3	使用命令对象	638
21.11	System.ComponentModel 命名空间 的作用	610	22.11	使用数据读取器	639
21.11.1	增强 CarControl 的设计时 外观	611	22.12	使用命令对象修改表	641
21.11.2	定义默认的属性和默认 事件	611	22.12.1	插入新的记录	643
21.11.3	指定自定义的工具箱位图	612	22.12.2	删除现有记录	643
21.12	创建自定义对话框	613	22.12.3	更新现有记录	643
21.12.1	DialogResult 属性	614	22.13	使用参数化的命令对象	644
21.12.2	窗体继承	615	22.14	使用 DbCommand 执行存储过程	646
21.13	动态定位 Windows 窗体控件	617	22.15	.NET 2.0 的异步数据访问	647
21.13.1	Anchor 属性	617	22.16	ADO.NET 断开式访问方式	648
21.13.2	Dock 属性	617	22.17	DataSet 的作用	649
21.13.3	表和流布局	618	22.18	使用 DataColumn	651
21.14	小结	619	22.18.1	构建 DataColumn	652
第 22 章	使用 ADO.NET 访问数据库	620	22.18.2	启用列自增	652
22.1	ADO.NET 高层次定义	620	22.18.3	把 DataColumn 加入 DataTable	652
22.2	ADO.NET 的数据提供者	621	22.19	使用 DataRow	653
22.2.1	微软提供的数据库提供者	622	22.20	使用 DataTable	654
22.2.2	选择第三方的数据库提供者	623	22.21	持久化 DataSet (和 DataTable) 成为 XML	657
22.3	其他的 ADO.NET 命名空间	623	22.22	把 DataTable 呈现到用户界面	658
22.4	System.Data 类型	624	22.22.1	以编程方式删除行	660
22.4.1	IDbConnection 接口的作用	624	22.22.2	应用过滤和排序	660
22.4.2	IDbTransaction 接口的作用	625	22.22.3	更新行	662
22.4.3	IDbCommand 接口的作用	625	22.23	使用 DataView 类型	663
22.4.4	IDbDataParameter/IData- Parameter 接口的作用	625	22.24	使用数据适配器	664
22.4.5	IDbDataAdapter/IData- Adapter 接口的作用	626	22.24.1	使用数据适配器填充 DataSet	665
22.4.6	IDataReader/IDataRecord 接口的作用	626	22.24.2	映射数据库名称为友好 名称	665
22.5	使用接口抽象数据提供者	627	22.25	使用数据适配器对象更新数据库	666
22.6	使用应用程序配置文件增加灵活性	628	22.25.1	设置 InsertCommand 属性	667
22.7	.NET 2.0 提供者工厂模型	629	22.25.2	设置 UpdateCommand 属性	667
22.7.1	为数据库提供者工厂注册	630	22.25.3	设置 DeleteCommand 属性	668
22.7.2	完整的数据提供者的例子	631	22.26	使用 CommandBuilder 类型自动 生成 SQL 命令	668
22.8	<connectionStrings> 元素	632	22.27	多表 DataSet 和 DataRelation 对象	670
22.9	安装 Cars 数据库	633	22.28	最后看一下 (数据) 向导	673
22.10	ADO.NET 的连接式访问	635	22.28.1	强类型化的 DataSet	675
22.10.1	使用连接对象	635	22.28.2	自动生成的数据组件	675

22.29	小结	676	作用	707	
第五部分 Web 应用程序和 XML Web 服务					
第 23 章 ASP.NET 2.0 网页和 Web 控件					
23.1	HTTP 的作用	678	23.15.2	Error 事件	708
23.2	Web 应用程序和 Web 服务	679	23.16	Web 控件的本质	709
23.2.1	使用 IIS 虚拟目录工作	680	23.16.1	取得服务器端事件处理权	710
23.2.2	ASP.NET 2.0 开发服务器	680	23.16.2	AutoPostBack 属性	710
23.3	HTML 的作用	681	23.17	System.Web.UI.Control 类型	711
23.3.1	HTML 文档结构	681	23.17.1	枚举所包含的控件	711
23.3.2	HTML 表单开发	682	23.17.2	动态添加 (和删除) 控件	712
23.3.3	构建基于 HTML 的用户界面	682	23.18	System.Web.UI.WebControls.WebControl 类型的关键成员	713
23.4	客户端脚本的作用	684	23.19	ASP.NET Web 控件的类别	714
23.4.1	客户端脚本示例	685	23.20	构建简单的 ASP.NET 2.0 站点	714
23.4.2	验证 default.htm 表单数据	685	23.20.1	使用母版页工作	715
23.5	提交表单数据 (GET 和 POST)	685	23.20.2	定义 Default.aspx 内容页面	717
23.6	构建传统的 ASP 页面	686	23.20.3	设计 Inventory 内容页面	718
23.7	传统 ASP 相关问题	688	23.20.4	设计 Build a Car 内容页面	722
23.7.1	ASP.NET 1.x 的主要优点	688	23.21	验证控件的作用	724
23.7.2	ASP.NET 2.0 的主要改进	688	23.21.1	RequiredFieldValidator	725
23.8	ASP.NET 2.0 命名空间	688	23.21.2	RegularExpressionValidator	725
23.9	ASP.NET 网页代码模型	689	23.21.3	RangeValidator	726
23.9.1	使用单文件页面模型	690	23.21.4	CompareValidator	726
23.9.2	使用代码隐藏页面模型	694	23.21.5	创建 ValidationSummary	727
23.10	ASP.NET 站点目录结构细节	697	23.22	小结	727
23.10.1	Bin 文件夹的作用	697	第 24 章 ASP.NET 2.0 Web 应用程序		728
23.10.2	App_Code 文件夹的作用	698	24.1	状态问题	728
23.11	ASP.NET 2.0 页面编译周期	699	24.2	ASP.NET 状态管理技术	730
23.11.1	单文件页面的编译周期	699	24.3	ASP.NET 视图状态的作用	730
23.11.2	多文件页面的编译周期	700	24.3.1	演示视图状态	730
23.12	页面类型的继承链	701	24.3.2	添加自定义视图状态数据	732
23.13	与传入的 HTTP 请求交互	702	24.3.3	控件状态简述	732
23.13.1	获得浏览器统计数据	703	24.4	Global.asax 文件的作用	733
23.13.2	访问传入的表单数据	703	24.4.1	全局最后异常事件处理程序	734
23.13.3	IsPostBack 属性	704	24.4.2	HttpApplication 基类	735
23.14	与输出 HTTP 响应交互	704	24.5	应用程序状态与会话状态差别	735
23.14.1	提交 HTML 内容	705	24.5.1	维护应用程序级的状态数据	736
23.14.2	重定向用户	705	24.5.2	修改应用程序数据	737
23.15	ASP.NET 网页的生命周期	706	24.5.3	处理 Web 应用程序的关闭	738
23.15.1	AutoEventWireUp 特性的		24.6	使用应用程序缓存	738
			24.6.1	使用数据缓存	739

24.6.2	修改*.aspx 文件	741	25.9.1	通过 Description 属性为 Web 方法归档	765
24.7	维护会话数据	743	25.9.2	通过 MessageName 属性避免 WSDL 名称冲突	766
24.8	cookie	745	25.9.3	用 EnableSession 属性构建有状态的 Web 服务	766
24.8.1	创建 cookie	746	25.10	探索 WSDL	768
24.8.2	读取传入的 cookie 数据	747	25.10.1	定义 WSDL 文档	768
24.9	使用 Web.config 配置 ASP.NET 应用程序	747	25.10.2	<types>元素	769
24.9.1	通过<trace>启用跟踪	748	25.10.3	<message>元素	770
24.9.2	通过<customErrors>自定义错误输出	749	25.10.4	<portType>元素	770
24.9.3	通过<sessionState>存储状态	750	25.10.5	<binding>元素	771
24.9.4	ASP.NET 2.0 站点管理工具	751	25.10.6	<service>元素	771
24.10	配置继承	752	25.11	再谈 XML Web 服务报文协议	772
24.11	小结	753	25.11.1	HTTP GET 和 HTTP POST 绑定	772
第 25 章 XML Web 服务		754	25.11.2	SOAP 绑定	773
25.1	XML Web 服务的作用	754	25.12	wsdl.exe 命令行的效用	774
25.1.1	XML Web 服务的优点	754	25.12.1	将 WSDL 转换成服务器端 XML Web 服务框架	775
25.1.2	定义 XML Web 服务客户端	755	25.12.2	将 WSDL 转换为客户端代理类	775
25.1.3	XML Web 服务的基础	755	25.13	查看代理服务器代码	776
25.1.4	概述 XML Web 服务发现	755	25.13.1	默认的构造函数	776
25.1.5	概述 XML Web 服务描述	756	25.13.2	同步调用支持	777
25.1.6	概述传输协议	756	25.13.3	异步调用支持	777
25.2	.NET XML Web 服务命名空间	756	25.13.4	构建客户端应用程序	778
25.3	手动构建 XML Web 服务	757	25.14	使用 Visual Studio 2005 生成代理类	778
25.3.1	使用 WebDev.WebServer.exe 测试 XML Web 服务	758	25.15	从 Web 方法公开自定义类型	779
25.3.2	使用 IIS 测试 XML Web 服务	759	25.15.1	公开数组	779
25.3.3	查看 WSDL 合约	759	25.15.2	公开结构	780
25.4	自动生成测试页面	759	25.15.3	公开 ADO.NET 数据集	781
25.5	使用 Visual Studio 2005 构建 XML Web 服务	760	25.15.4	Windows 窗体客户端	781
25.6	WebService 基类的作用	762	25.15.5	客户端类型代理	783
25.7	[WebService]特性	763	25.16	发现服务协议 (UDDI)	784
25.7.1	Namespace 和 Description 属性的作用	763	25.17	小结	785
25.7.2	Name 属性	763	第六部分 .NET 3.0 扩展编程		
25.8	[WebServiceBinding]特性	764	第 26 章 建立.NET 3.0 编程环境		
25.8.1	忽略 BP 1.1 一致性验证	765	26.1	.NET 3.0 技术介绍	788
25.8.2	禁用 BP 1.1 一致性验证	765	26.2	C# 3.0 和 LINQ 技术介绍	789
25.9	[WebMethod]特性	765			

26.3	欢迎使用.NET 3.0	789	27.10.2	在 WrapPanel 面板中放置内容	820
26.4	安装.NET Framework 3.0 运行库组件	790	27.10.3	在 StackPanel 面板内放置内容	821
26.5	安装 Windows 软件开发包	791	27.10.4	在 Grid 面板中放置内容	822
26.5.1	选择安装项	791	27.10.5	在 DockPanel 面板中放置内容	823
26.5.2	研究 SDK 的内容	792	27.10.6	使用嵌套的面板创建窗体的框架	823
26.6	安装 Visual Studio “Orcas” 开发工具	793	27.11	WPF 控件	825
26.6.1	安装 WPF 和 WCF 项目支持	793	27.11.1	配置 WPF 控件	825
26.6.2	安装 Visual Studio 2005 为 WF 提供的扩展	794	27.11.2	使用 WPF 控件属性	826
26.7	安装 C# 3.0 和 LINQ 社区预览版	795	27.11.3	处理 WPF 控件事件	828
26.8	小结	796	27.11.4	应用控件样式	829
第 27 章	WPF 介绍	797	27.12	WPF 图形显示服务简介	831
27.1	WPF 背后的动机	797	27.12.1	WPF 图形服务详解	832
27.1.1	通过 XAML 将关注点 分离	798	27.12.2	使用基本的形状	833
27.1.2	提供优化的呈现模型	799	27.12.3	WPF 动画服务介绍	834
27.2	WPF 程序集详解	799	27.12.4	使用微软 Expression 图形 设计器生成 XAML	836
27.2.1	Application 类的作用	800	27.13	XAML 浏览器应用程序简介	836
27.2.2	Window 类的作用	801	27.14	小结	838
27.3	创建（不使用 XAML 的）WPF 应用程序	803	第 28 章	WCF 介绍	839
27.3.1	扩展 Window 类	805	28.1	WCF 背后的动机	839
27.3.2	创建简单的用户界面	805	28.2	探究 WCF 核心程序集	840
27.4	XAML 介绍	806	28.3	WCF 基础	841
27.4.1	用 XAML 定义 MainWindow	807	28.3.1	WCF 契约	841
27.4.2	用 XAML 定义应用对象	808	28.3.2	WCF 绑定	841
27.4.3	通过 msbuild.exe 处理 XAML 文件	809	28.3.3	WCF 地址	843
27.5	将标记转换为.NET 程序集	810	28.4	构建完整的 WCF 应用程序	843
27.5.1	XAML 到 C#代码的映射	810	28.4.1	组成 WCF 应用程序的相关 程序集	843
27.5.2	BAML 的作用	811	28.4.2	契约的定义与实现	843
27.5.3	XAML 到程序集的过程摘要	813	28.5	承载 WCF 服务	846
27.6	使用代码隐藏文件实现的关注点 的分离	813	28.5.1	指明 ABC	848
27.7	在 XamlPad 中练习使用 XAML	815	28.5.2	ServiceHost 类型的功能	848
27.8	使用 Visual Studio “Orcas” 创建 WPF 应用程序	816	28.6	<system.ServiceModel>元素的细节	849
27.9	使用微软表达式交互设计器生成 XAML	817	28.7	与 WCF 服务进行通信	852
27.10	使用面板控制内容布局	818	28.7.1	使用 svcutil.exe 生成代理 代码	852
27.10.1	在 Canvas 面板中放置内容	819	28.7.2	使用 Visual Studio 2005 生成 代理代码	853

28.8	WCF 的数据类型表示	854	30.3.2	在实例层次上调用扩展方法	886
28.8.1	更新 ICarOrder 服务契约	854	30.3.3	静态调用扩展方法	887
28.8.2	对 CarOrderServiceClient 程序集重新编码	856	30.3.4	导入定义了扩展方法的类型	888
28.8.3	使用 XmlSerializer 进行数据 编码	857	30.3.5	构建和使用扩展库	888
28.8.4	使用二进制格式传输数据	857	30.4	理解对象初始化器	890
28.9	使用服务配置编辑器生成 WCF 配置文件	858	30.4.1	使用初始化语法调用自定义 构造函数	891
28.10	小结	859	30.4.2	初始化内部类型	892
第 29 章	WF 介绍	860	30.4.3	理解集合的初始化	893
29.1	WF 背后的动机	860	30.5	理解匿名类型	894
29.2	WF 的积木块	861	30.5.1	匿名类型的内部表示方式	895
29.2.1	WF 中的集成服务	862	30.5.2	方法 ToString() 和方法 GetHashCode() 的实现	896
29.2.2	WF 活动初览	862	30.5.3	匿名类型的相等语义	897
29.2.3	顺序工作流和状态机工作流 的作用	863	30.5.4	包含匿名类型的匿名类型	898
29.2.4	深入探讨工作流	865	30.6	理解 Lambda 表达式的角色	899
29.3	WF 程序集和核心命名空间	865	30.6.1	Lambda 表达式是更好的匿 名方法	900
29.4	建造一个启用工作流的简单应用	866	30.6.2	剖析 Lambda 表达式	902
29.4.1	研究初始工作流的代码	866	30.6.3	Lambda 表达式的两种风格	902
29.4.2	添加 Code 活动	867	30.6.4	使用 Lambda 表达式重新编 写 CarDelegate 示例	903
29.4.3	添加 While 活动	868	30.6.5	含有多个 (或零个) 参数的 Lambda 表达式	905
29.4.4	研究 WF 引擎承载代码	870	30.7	小结	906
29.4.5	添加定制的起初参数	871	第 31 章	LINQ 介绍	907
29.5	在工作流中调用 Web 服务	873	31.1	定义 LINQ 的作用	907
29.6	构建可重用的 WF 代码库	877	31.2	核心 LINQ 程序集	908
29.6.1	编写简单的工作流	878	31.3	LINQ 查询表达式初览	909
29.6.2	创建启用工作流的 Windows Forms 应用程序	878	31.3.1	重访隐型局部变量	911
29.7	关于自定义活动的简要说明	879	31.3.2	重访扩展方法	912
29.8	小结	880	31.4	用 LINQ 查询泛型集合	913
第 30 章	C# 3.0 的语言功能	881	31.4.1	定义 LINQ 查询	914
30.1	使用 C# 3.0 命令行编译器	881	31.4.2	重访匿名类型	914
30.2	理解隐式类型化的局部变量	882	31.5	使用 LINQ 查询非泛型集合	915
30.2.1	隐式类型化变量的限制	883	31.6	查询运算符的内部表示	916
30.2.2	隐式类型化的局部数组	884	31.6.1	用查询运算符建立查询表达 式 (复习)	917
30.2.3	隐式数据类型化的最后注意 事项	884	31.6.2	使用 Sequence 类型和 Lambda 表达式来建立查询表达式	917
30.3	理解扩展方法	885	31.6.3	使用 Sequence 类型和匿名方 法来建立查询表达式	918
30.3.1	定义扩展方法	885			

31.6.4	用 Sequence 类型和原始代理 建立查询表达式.....	919	31.10.1	研究生成的实体类.....	930
31.7	研究 LINQ 查询运算符.....	920	31.10.2	使用实体类来定义关系.....	931
31.8	构建 LINQ 查询表达式.....	920	31.10.3	强类型的 DataContext.....	932
31.8.1	基本的选择语法.....	921	31.10.4	针对生成的类型来编程.....	933
31.8.2	获取数据子集.....	922	31.11	使用 Visual Studio 2005 建立实体类.....	934
31.8.3	逆转结果集的顺序.....	923	31.11.1	插入新项.....	935
31.8.4	对表达式进行排序.....	923	31.11.2	更新现有项.....	936
31.8.5	转换查询结果以及转换延缓 执行的作用.....	924	31.11.3	删除现有项.....	936
31.9	使用 LINQ 到 SQL 来查询关系 数据库.....	925	31.12	使用 LINQ 到 XML 操作 XML 文档.....	936
31.9.1	实体类的作用.....	926	31.12.1	System.Xml.Linq 命名 空间.....	937
31.9.2	DataContext 类型的作用.....	926	31.12.2	以编程方式创建 XML 文档.....	937
31.9.3	一个 LINQ 到 SQL 的简单 例子.....	926	31.12.3	装载并分析 XML 内容.....	939
31.9.4	建立强类型的 DataContext.....	927	31.13	在内存文档中导航.....	940
31.9.5	详细介绍 [Table] 特性和 [Column] 特性.....	928	31.13.1	使用 LINQ 到 XML 来 选择元素.....	940
31.10	使用 sqlmetal.exe 生成实体类.....	929	31.13.2	在 XML 文档中修改数据.....	941
			31.14	小结.....	942