

<<C#和.NET 2.0实战>>

图书基本信息

书名：<<C#和.NET 2.0实战>>

13位ISBN编号：9787115166203

10位ISBN编号：711516620X

出版时间：2008-1

出版时间：人民邮电

作者：Patrick Smacchia

页数：765

译者：施凡

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<C#和.NET 2.0实战>>

内容概要

本书是一本知识密集的C#技术经典图书，Microsoft .NET MVP力作，众多.NET专家口口相传的一本秘籍。

全书分为三个部分，第一部分讲述底层的.NET平台，涵盖了.NET各方面的基础知识和工作原理；第二部分是C#语言部分，通过与C++比较的方式进行讲解，清晰易懂；第三部分讲述.NETFramework中的基本类库，内容几乎涉及.NET常见领域的全部知识。

本书主要面向熟悉.NET的编程人员，也适合.NET技术的初学者阅读。

作者简介

来自法国的微软MVP，拥有多年大型项目开发经验。

书籍目录

第1章 .NET简介1.1 什么是.NET1.1.1 微软软件开发平台1.1.2 一组规范1.1.3 .NET概览1.2 发展历程1.2.1 过去1.2.2 现在1.2.3 未来1.3 微软和Windows以外的.NET1.3.1 ECMA组织与.NET1.3.2 W3C联盟1.3.3 Mono项目1.3.4 微软SSCLI项目1.4 .NET资源链接1.4.1 网站1.4.2 新闻组1.4.3 博客第一部分 .NET平台第2章 程序集、模块和IL语言2.1 程序集、模块和资源文件2.1.1 程序集和模块2.1.2 资源文件2.1.3 程序集、模块、类型和资源2.1.4 为何对多模块程序集感兴趣2.1.5 ILMerge工具2.2 模块的剖析2.2.1 可移植的可执行文件简介2.2.2 模块的结构2.2.3 清单的结构2.2.4 类型元数据段的结构2.3 使用ildasm.exe和Reflector工具分析程序集2.3.1 创建需要分析的程序集2.3.2 使用ildasm.exe分析模块2.3.3 Reflector工具2.4 程序集attribute和版本设定2.4.1 程序集的标准attribute2.4.2 程序集的版本设定2.4.3 友元程序集2.5 强名称程序集2.5.1 简介2.5.2 sn.exe工具2.5.3 公钥记号2.5.4 为程序集签名2.5.5 具体示例2.5.6 程序集的延迟签名2.6 国际化/本地化与卫星程序集2.6.1 区域设置和本地化2.6.2 资源文件2.6.3 在代码中使用资源2.6.4 创建卫星程序集2.6.5 部署和使用卫星程序集2.6.6 避免在资源无法找到时引发异常2.6.7 Visual Studio与卫星程序集2.6.8 区域设置与字符串格式化2.7 IL语言简介2.7.1 栈及其特殊的IL指令2.7.2 示例1：局部变量与栈2.7.3 示例2：方法调用与栈2.7.4 用于比较、分支和跳转的IL指令2.7.5 IL的面向对象特性2.7.6 元数据符号第3章 生成、部署以及配置.NET应用程序3.1 用MSBuild生成应用程序3.2 MSBuild：目标、任务、属性、项与条件3.2.1 .proj文件、目标与任务3.2.2 属性3.2.3 项3.2.4 条件3.3 高级MSBuild3.3.1 增量生成与目标间的依赖3.3.2 MSBuild转换3.3.3 将一个MSBuild项目分解到多个文件3.3.4 Visual Studio 2005如何利用MSBuild3.3.5 创建自定义MSBuild任务3.4 配置文件3.4.1 machine.config文件3.4.2 标准配置参数3.4.3 使用 appSettings 元素定义配置参数3.4.4 使用配置节定义配置参数3.4.5 使用Visual Studio 2005创建配置节3.4.6 配置节的注意事项3.5 程序集部署：XCopy与GAC3.5.1 XCopy部署3.5.2 共享程序集与GAC文件夹3.5.3 GAC的并存存储模型是如何解决DLL hell问题的3.5.4 并存执行3.5.5 查看及编辑GAC文件夹3.6 发布者策略程序集3.6.1 潜在的问题3.6.2 解决方案3.6.3 创建发布者策略程序集3.7 .NET应用程序部署简介3.7.1 MSI、cab、XCopy、ClickOnce和NTD之间的对比3.7.2 MSI与ClickOnce的对比3.8 使用cab文件部署应用程序3.9 使用MSI技术部署应用程序3.9.1 添加文件3.9.2 安装快捷方式3.9.3 在GAC文件夹中添加一个共享程序集3.9.4 安装项目属性3.9.5 更新注册表3.9.6 指定在安装期间执行的自定义动作3.9.7 为安装提供一个自定义用户界面3.10 使用ClickOnce技术部署应用程序3.10.1 部署文件夹3.10.2 为ClickOnce部署做准备3.10.3 ClickOnce部署与移动代码安全3.10.4 按需安装与下载组3.10.5 更新一个使用ClickOnce安装的应用程序3.10.6 应用程序所需的CAS权限集的工作机制3.10.7 ClickOnce应用程序安装与执行的细节3.11 使用无接触部署(NTD)技术部署应用程序3.12 如果目标机器上没有安装.NET运行库怎么办第4章 CLR4.1 应用程序域4.1.1 简介4.1.2 线程与AppDomain4.1.3 卸载AppDomain4.1.4 AppDomain和孤立性4.1.5 System.AppDomain类4.1.6 在一个进程中承载多个应用程序4.1.7 在其他AppDomain的上下文中运行代码4.1.8 AppDomain类的事件4.1.9 在同一个进程的AppDomain之间共享信息4.2 在Windows进程内通过运行库宿主加载CLR4.2.1 mscorsvr.dll和mscorwks.dll4.2.2 mscorlib.dll程序集4.2.3 运行库宿主介绍4.2.4 在同一台计算机上承载多个版本的CLR4.2.5 使用CorBindToRuntimeExO函数加载CLR4.2.6 创建一个自定义的运行库宿主4.2.7 在自定义运行库宿主中调整CLR4.2.8 SQL Server 2005运行库宿主的特性4.3 剖析.NET应用程序的执行状况4.4 定位和加载程序集4.4.1 CLR何时尝试定位程序集4.4.2 CLR使用的定位算法4.4.3 配置文件的 assemblyBinding 元素4.4.4 定位算法示意图4.4.5 影子复制机制4.5 运行库类型解析4.5.1 显式或隐式加载程序集4.5.2 编译时引用程序集4.5.3 示例4.5.4 类型解析算法示意图4.6 JIT(即时)编译4.6.1 可移植的二进制代码4.6.2 即时编译技术简介4.6.3 ngen.exe工具4.6.4 性能计数器与JIT编译4.7 垃圾收集器和托管堆4.7.1 垃圾收集技术简介4.7.2 垃圾收集算法遇到的问题4.7.3 .NET的GC4.7.4 第一步：寻找根对象4.7.5 第二步：建立活动对象树4.7.6 第三步：解除分配非活动对象4.7.7 第四步：清理堆碎片4.7.8 第五步：重新计算托管引用所使用的物理地址4.7.9 推荐做法4.7.10 针对大对象的特殊堆4.7.11 多线程环境下的垃圾收集4.7.12 弱引用4.7.13 使用System.GC

类影响GC的行为4.8 提高代码可靠性的机制4.8.1 异步异常及托管代码可靠性4.8.2 受约束执行区域4.8.3 如何定义CER4.8.4 内存门4.8.5 可靠性契约4.8.6 关键终结器4.8.7 临界区4.9 CLI和CLS4.9.1 .NET语言必须满足的要求4.9.2 从开发人员的观点看CLI和CLS第5章 进程、线程与同步5.1 简介5.2 进程5.2.1 简介5.2.2 System.Diagnostics.Process类5.2.3 创建和销毁子进程5.2.4 避免在一台机器上同时运行同一应用程序的多个实例5.2.5 终止当前进程5.3 线程5.3.1 简介5.3.2 受托管的线程与Windows线程5.3.3 抢占式多任务处理5.3.4 进程与线程的优先级5.3.5 System.Threading.Thread类5.3.6 创建与联结线程5.3.7 挂起线程5.3.8 终止线程5.3.9 前台线程与后台线程5.3.10 受托管线程的状态图5.4 访问资源同步简介5.4.1 竞态条件5.4.2 死锁5.5 使用volatile字段与Interlocked类实现同步5.5.1 volatile字段5.5.2 System.Threading.Inter-locked类5.5.3 Interlocked类提供的其他功能5.6 使用System.Threading.Monitor类与C#的lock关键字实现同步5.6.1 EnterO方法和ExitO方法5.6.2 C#的lock关键字5.6.3 SyncRoot模式5.6.4 线程安全类5.6.5 Monitor.TryEnterO方法5.6.6 Monitor类的WaitO方法、PulseO方法以及PulseAllO方法5.7 使用Win32对象同步：互斥体、事件与信号量5.7.1 共享Win32同步对象5.7.2 互斥体5.7.3 事件5.7.4 信号量5.8 利用System.Threading.Reader-WriterLock类实现同步 1135.9 利用System.Runtime.Remoting.Contexts.SynchronizationAttribute实现同步5.9.1 同步域简介5.9.2 System.Runtime.Remoting.Contexts.Synchronization与同步域5.9.3 重入与同步域5.9.4 另一个名为Synchronization的attribute5.10 CLR的线程池5.10.1 简介5.10.2 使用线程池5.11 定时器5.11.1 System.Timers.Timer类5.11.2 System.Threading.Timer类5.11.3 System.Windows.Forms.Timer类5.12 异步方法调用5.12.1 异步委托5.12.2 回调方法5.12.3 向回调方法传递状态5.12.4 one-way调用5.13 线程-资源亲缘性5.13.1 System.ThreadStatic-Attribute5.13.2 线程本地存储(TLS)5.13.3 System.ComponentModel.ISynchronizeInvoke接口5.14 执行上下文简介第6章 安全性6.1 代码访问安全性(CAS)概述6.1.1 什么是移动代码6.1.2 CAS：全局观6.1.3 给程序集代码授予权限6.1.4 在运行程序集的代码时检查权限6.2 CAS：证据和权限6.2.1 什么是证据6.2.2 .NET Framework所提供的标准证据6.2.3 谁提供证据6.2.4 权限6.2.5 标准权限6.2.6 标识权限6.2.7 安全权限6.2.8 自定义权限6.3 CAS：通过应用安全策略根据证据授予权限6.3.1 安全策略级别6.3.2 剖析安全策略6.3.3 用于应用安全策略的算法6.3.4 默认安全策略配置6.3.5 配置安全策略6.4 CAS：FullTrust权限6.5 CAS：从源代码进行命令式的权限检查6.5.1 CodeAccessPermissions类和PermissionSet类6.5.2 DemandO方法6.5.3 DenyO方法、RevertDenyO方法、PermitOnlyO方法和RevertPermitOnlyO方法6.5.4 AssertO方法和RevertAssertO方法6.5.5 FromXmlO方法和ToXmlO方法6.5.6 System.Security.IPermission接口6.6 CAS：使用attribute进行声明式的权限检查6.6.1 加载程序集时调整所授权限集合的attribute6.6.2 命令式与声明式的对比6.7 CAS：测试和调试移动代码的实用机制6.8 CAS：独立存储区权限6.9 .NET、Windows用户与角色6.9.1 Windows安全简介6.9.2 IIdentity接口与IPrincipal接口6.9.3 Windows安全标识符6.9.4 在底层Windows线程中模拟用户6.10 .NET与Windows资源的访问控制6.10.1 Windows访问控制简介6.10.2 在.NET代码中使用特殊的SD6.10.3 在.NET代码中使用通用的SD6.11 .NET与角色6.11.1 定义应用程序域的主体策略6.11.2 检查用户是否属于某个特定角色6.11.3 COM+角色6.12 .NET与密码学：对称算法6.12.1 对称算法概述6.12.2 .NET Framework与对称算法6.13 .NET与密码学：非对称算法(公钥/私钥)6.13.1 非对称算法概述6.13.2 安全会话简介6.13.3 RSA算法6.13.4 非对称算法与数字签名6.13.5 .NET Framework与RSA算法6.14 数据保护API6.14.1 Windows的数据保护API6.14.2 System.Security.Cryptography.ProtectedData类6.14.3 System.Security.Cryptography.ProtectedMemory类6.14.4 System.Security.SecureString类6.14.5 保护配置文件中的数据6.14.6 确保网络传输中数据的安全6.15 使用Authenticode技术与X.509证书验证程序集6.15.1 Authenticode与强名称6.15.2 证书与证书认证中心6.15.3 根证书6.15.4 Windows、.NET与Authenticode技术第7章 反射、后期绑定与attribute7.1 反射7.1.1 何时需要反射7.1.2 .NET反射有何新意7.1.3 对载入AppDomain的程序集的反射7.1.4 从元数据获取信息7.2 后期绑定7.2.1 “绑定类”的含义7.2.2 早期绑定与动态绑定7.2.3 后期绑定7.2.4 在C#编译到IL期间如何实例化一个未知的类7.2.5 使用后期绑定7.2.6 利用接口：使用后期绑定的正确方法7.3 attribute7.3.1 attribute是什么7.3.2 何时需要attribute7.3.3 关于attribute应该知道的事7.3.4 可以应用attribute的代码元素7.3.5

.NET Framework中的一些标准attribute7.3.6 自定义的attribute的示例7.3.7 条件attribute7.4 动态生成程序集并在运行中使用7.4.1 为什么要考虑动态生成程序集7.4.2 一个实际的问题 1867.4.3 理想的第三种解决方案——动态创建程序集7.4.4 将程序集保存到磁盘上的能力7.4.5 结论第8章 .NET与本机代码/COM/COM+之间的互操作性8.1 P/Invoke8.1.1 DllImportAttribute8.1.2 类型转换8.1.3 使用指针传递参数8.1.4 传递字符串8.1.5 传递结构与联合8.1.6 方向attribute8.1.7 委托与非托管的函数指针8.2 C++/CLI语言互操作性简介8.2.1 IJW功能8.2.2 托管类型与非托管类型共存8.2.3 在本机代码中使用托管对象8.3 .NET与Win32句柄8.3.1 简介8.3.2 HandleCollector类8.3.3 SafeHandle类与Critical-Handle类8.4 在.NET中使用COM对象8.4.1 类型元数据与类型库8.4.2 Interop程序集与Runtime Callable Wrapper类8.4.3 访问不使用类型库的COM组件中的COM类8.4.4 使用Visual Studio导入ActiveX8.4.5 使用RCW类时需要牢记的COM特性8.4.6 COM类的显式后期绑定8.4.7 免注册COM8.5 将.NET对象包装成CCW8.5.1 CCW简介8.5.2 从.NET程序集中生成描述CCW类的COM类型库8.5.3 在操作系统上注册CCW8.5.4 将.NET程序集作为COM组件使用8.5.5 .NET异常与CCW8.5.6 处理对象生命周期8.6 COM+简介8.6.1 COM+是什么8.6.2 在.NET中使用COM+：企业服务7.7 COM+企业服务简介8.7.1 COM+企业服务的完整列表8.7.2 你需要COM+企业服务吗8.8 在.NET类中利用COM+服务8.8.1 服务组件简介8.8.2 声明服务组件使用的企业服务8.8.3 COM+上下文与服务组件8.8.4 COM+应用程序简介8.8.5 COM+目录8.8.6 COM+应用程序的激活模式8.8.7 安装服务组件8.8.8 查看并操作COM+目录8.8.9 构建一个使用服务组件的客户第二部分 C#语言第9章 语言基本概念9.1 源代码组织9.1.1 命名空间9.1.2 命名空间中可声明的全套语法元素9.1.3 C#项目的结构9.1.4 命名空间和程序集的组织9.2 编译步骤9.3 预处理器9.3.1 预处理符号与条件编译9.3.2 预处理符号与条件attribute9.3.3 #error指令和#warning指令9.3.4 #pragma warning disable 指令和#pragma warning restore指令9.3.5 #line指令9.3.6 #region指令和#endregion指令9.4 csc.exe编译器9.5 别名机制9.5.1 创建命名空间和类型的别名9.5.2 命名空间别名限定符9.5.3 全局限定符9.5.4 外部别名9.6 注释和自动文档9.6.1 注释9.6.2 使用Visual Studio的任务列表9.6.3 自动文档9.7 标识符9.8 控制结构9.8.1 条件语句(if/else、?:和switch)9.8.2 循环语句(do、while、for和foreach)9.8.3 goto语句9.9 MainO方法第10章 从C# 2.0的角度看.NET 2.0类型系统10.1 对象在内存中的存储10.1.1 对象的分配和解除分配10.1.2 托管线程栈10.1.3 托管堆10.1.4 对象存储：托管线程栈与托管堆的比较10.1.5 静态分配与动态分配的比较10.2 引用类型和值类型10.3 公共类型系统(CTS)10.3.1 .NET类型不与任何编程语言关联10.3.2 CTS：全局观10.4 System.Object类10.5 对象比较10.5.1 等价与恒等10.5.2 自定义相等比较规则10.5.3 相等性与散列表10.5.4 自定义大小比较规则10.6 对象克隆10.7 装箱和拆箱10.7.1 装箱10.7.2 装箱导致的危险问题10.7.3 拆箱10.8 基本类型10.8.1 整型10.8.2 实型10.8.3 布尔型10.8.4 字符型10.8.5 整数和字符串之间的转换10.9 基本类型的运算10.9.1 同一基本类型上的算术运算10.9.2 处理被零除的错误10.9.3 处理溢出10.9.4 运算符的优先级10.9.5 前缀和后缀递增/递减运算符10.9.6 不同基本类型之间的算术运算10.9.7 位运算10.10 结构10.11 枚举10.11.1 枚举和整型10.11.2 System.Enum类10.11.3 位域(一组标志)10.12 字符串10.12.1 System.String类10.12.2 字符串字面常量10.12.3 无转义字符串字面常量10.12.4 使用字符串10.12.5 格式化字符串10.12.6 System.Text.StringBuilder类10.13 委托类和委托对象10.13.1 简介10.13.2 委托对象与静态方法10.13.3 C# 2.0编译器的委托类推测功能10.13.4 委托对象与实例方法10.13.5 用一个委托对象引用多个方法10.13.6 System.Delegate类10.13.7 更改引用方法列表10.14 可空类型10.14.1 值类型和空值范式10.14.2 System.Nullable T结构10.14.3 C#语法革新：Nullable T关键字和null关键字10.14.4 C#语法革新：Nullable T的等价写法T?10.14.5 C# 2.0中bool?的无差别对待10.14.6 可空类型与装箱/拆箱10.14.7 可空结构和可空枚举10.15 部分类型10.15.1 必须在每个部分声明上重复出现的类型声明元素10.15.2 在每个部分声明上可选重复出现的类型声明元素10.15.3 在多个部分声明上重复出现就会累积效果的类型声明元素10.15.4 仅在某个部分声明上起作用的类型声明元素第11章 类和对象11.1 简介11.2 术语11.3 类的定义11.4 字段11.4.1 字段初始化11.4.2 常数字段11.4.3 字段初始化的潜在问题11.5 方法11.5.1 按值或按引用传递参数11.5.2 C#的默认规则11.5.3 按引用传递任意参数11.5.4 在引用类型上使用ref关键字11.5.5 参数初始化11.5.6 输出参数11.5.7 params关键字11.5.8 方法重载11.6 属性11.6.1 get访问器11.6.2 set访问器11.6.3 关于属性

的注释11.7 索引器11.8 事件11.8.1 介绍11.8.2 C#语法11.8.3 一个实际的例子11.8.4 异步事件处理11.8.5 在同步条件下保护代码不受订阅者方法所抛异常的影响11.9 嵌套类型11.10 封装性和可见性11.10.1 成员的可见性11.10.2 类型的可见性11.10.3 属性和索引器访问器的可见性11.10.4 私有成员的附加说明11.10.5 IL语言可见性的附加说明11.11 this关键字11.12 构造函数11.12.1 构造函数的声明11.12.2 创建对象时访问构造函数11.13 对象终结和析构11.13.1 析构函数、终结器和Object.FinalizeO方法11.13.2 IDisposable接口和DisposeO方法11.13.3 何时需要终结器和DisposeO方法11.14 静态成员11.14.1 静态字段、属性和事件11.14.2 静态方法11.14.3 静态构造函数11.14.4 静态类11.15 运算符重载11.15.1 算术运算符的重载11.15.2 类型转换运算符的重载11.15.3 比较运算符的重载11.15.4 运算符重载和CLS第12章 继承、多态性与抽象性12.1 目标：代码重用12.1.1 潜在问题12.1.2 解决方案之一：类的继承12.2 类的继承12.2.1 语法12.2.2 保护及内部或保护可见性级别12.2.3 类继承图12.2.4 调用基类的构造函数12.2.5 基类成员的可见性12.2.6 封闭类12.3 虚方法和多态性12.3.1 潜在问题12.3.2 解决方案之一：虚方法和多态性12.3.3 示例一则12.3.4 屏蔽多态性12.4 抽象性12.4.1 潜在的问题12.4.2 解决方案之一：抽象类和抽象方法12.4.3 示例一则12.4.4 同时使用abstract关键字和override关键字12.5 接口12.5.1 强制代码使用者使用抽象体代替实现12.5.2 解决方法命名的冲突12.5.3 解决接口扩展带来的冲突12.5.4 覆写接口的实现方法12.5.5 接口与结构12.6 虚拟或抽象属性、事件和索引器12.7 is和as运算符12.7.1 is运算符12.7.2 as运算符12.8 代码重用的解决方案第13章 泛型13.1 C# 1.0的一个问题以及使用C# 2.0泛型的解决方法13.1.1 C# 1.0中集合项的类型化问题13.1.2 C# 2.0泛型带来的理想方案13.2 .NET 2.0泛型概览13.2.1 声明多个类型参数13.2.2 开放和封闭泛型类型13.2.3 .NET泛型与C++模板的比较13.2.4 泛型类型的可见性13.2.5 泛型结构与接口13.2.6 泛型类型与别名13.3 类型参数的约束13.3.1 默认构造函数约束13.3.2 派生约束13.3.3 引用类型或值类型约束13.4 泛型类型的成员13.4.1 方法重载13.4.2 静态字段13.4.3 静态方法13.4.4 类构造函数13.4.5 运算符重载13.4.6 嵌套类型13.5 运算符与泛型13.5.1 类型参数的相等、不等及比较运算符13.5.2 typeof运算符与泛型13.5.3 params及lock关键字与泛型13.5.4 default运算符13.6 类型转换与泛型13.6.1 基本规则13.6.2 类型转换与泛型数组13.6.3 is和as运算符13.7 继承与泛型13.7.1 基本规则13.7.2 覆写泛型类型的虚方法13.8 泛型方法13.8.1 简介13.8.2 泛型方法与约束13.8.3 虚拟泛型方法13.8.4 泛型方法类型参数的推测13.8.5 C# 2.0语法的二义性13.9 委托、事件与泛型13.9.1 简介13.9.2 泛型委托与泛型方法13.9.3 泛型与委托的协变性及反变性13.9.4 事件和泛型委托13.10 反射、attribute、IL与泛型13.10.1 泛型与System.Type类13.10.2 泛型与System.Reflection.MethodBase类及System.Reflection.MethodInfo类13.10.3 attribute与泛型13.10.4 IL语言与泛型13.11 .NET Framework 2.0中的泛型13.11.1 对象序列化与泛型13.11.2 .NET Remoting与泛型13.11.3 集合与泛型13.11.4 不支持泛型的领域第14章 不安全代码、异常、匿名方法和迭代器14.1 指针和不安全代码14.1.1 允许不安全代码的编译器选项14.1.2 在C#中声明不安全代码14.2 在C#中使用指针14.2.1 .NET中支持指针的类型14.2.2 声明指针14.2.3 提领和间接运算符14.2.4 取大小运算符14.2.5 指针运算14.2.6 指针类型转换14.2.7 双重指针14.2.8 定址对象14.2.9 指针和数组14.2.10 固定数组14.2.11 使用stackalloc关键字在栈上分配内存14.2.12 字符串与指针14.3 利用异常处理错误14.3.1 潜在问题：如何恰当处理运行时出现的大多数错误14.3.2 C#异常处理简介14.4 异常对象和定义自己的异常类14.4.1 System.Exception类14.4.2 定义自己的异常类14.4.3 从自己的代码中抛出异常14.4.4 C#的非检查型异常14.5 catch块和finally块14.5.1 catch块的一些说明(异常处理程序)14.5.2 finally块14.5.3 增加异常语义14.6 从构造函数或终结器中抛出的异常14.6.1 静态构造函数所抛异常或静态字段初始化时所抛异常14.6.2 终结器所抛异常14.7 异常处理与CLR14.8 异常处理与Visual Studio14.9 异常管理使用指南14.9.1 何时考虑抛出异常14.9.2 在异常处理程序中做什么14.9.3 在何处放置异常处理程序14.9.4 异常与返回错误代码的对比14.9.5 永远不要预测bug造成的后果能够被异常处理程序所捕获14.10 匿名方法14.10.1 C# 2.0匿名方法简介14.10.2 匿名方法可以接受参数14.10.3 一个精妙的特殊语法14.10.4 匿名方法与泛型14.10.5 匿名方法的实际应用14.11 C# 2.0编译器与匿名方法14.11.1 简单的方式14.11.2 捕获局部变量14.11.3 捕获的局部变量与代码复杂性14.11.4 匿名方法访问外围方法的参数14.11.5 匿名方法访问外围类的成员14.12 匿名方法的高级用法14.12.1 定义：闭包与词法环境14.12.2 漫谈闭包14.12.3 使用闭包代替类14.12.4 委托与闭

包14.12.5 使用匿名方法处理集合14.13 C# 1.x迭代器14.13.1 可枚举对象、枚举器和Iterator设计模式14.13.2 实例一则14.13.3 一个可枚举对象对应多个枚举器14.13.4 C# 1.x迭代器的缺点14.14 C# 2.0迭代器14.14.1 yield return关键字14.14.2 迭代器与泛型14.14.3 一个可枚举对象对应多个枚举器14.14.4 yield break关键字14.14.5 yield return关键字和yield break关键字的语法限制14.14.6 递归迭代器实例14.15 C# 2.0编译器与迭代器14.15.1 编译器自动创建并使用的枚举器类14.15.2 关于生成类的说明14.16 C# 2.0迭代器的高级用法14.16.1 定义：协同例程和延拓14.16.2 通过迭代器利用协同例程和延拓的威力14.16.3 Pipeline模式14.16.4 延拓与多线程的比较14.16.5 C# 2.0迭代器的一个限制第三部分 .NET Framework第15章 集合15.1 使用foreach和in关键字迭代访问集合中的项15.1.1 在数组上使用foreach和in15.1.2 交错数组上的foreach语法15.1.3 在自定义集合类上支持foreach语法15.2 数组15.2.1 创建和引用数组15.2.2 访问数组的项和处理越界访问15.2.3 交错数组15.2.4 初始化数组的项15.2.5 数组的协变15.2.6 System.Array类15.2.7 位数组15.3 序列15.3.1 System.Collections.Generic.ICollection T 接口15.3.2 System.Collections.Generic.IList T 接口15.3.3 System.Collections.Generic.List T 类15.3.4 System.ComponentModel.IBindingList接口和System.ComponentModel.IListSource接口15.3.5 System.Collections.Generic.LinkedList T 类15.3.6 字符串列表15.3.7 System.Collections.Generic.Queue T 类15.3.8 System.Collections.Generic.Stack T 类15.4 字典15.4.1 System.Collections.Generic.IDictionary K,V 接口15.4.2 System.Collections.Generic.SortedDictionary K,V 类15.4.3 System.Collections.Generic.Dictionary K,V 类15.4.4 迭代访问字典中的项15.5 对集合中的项进行排序15.5.1 IComparer T 接口和IComparable T 接口15.5.2 对数组中的项排序15.5.3 对List T 实例中的元素排序15.6 用于处理集合的函数对象15.6.1 特化的委托类型15.6.2 使用匿名方法15.6.3 List T 类和Array类支持函数对象15.6.4 C# 2.0的迭代器和集合15.7 System.Collections.Generic和System.Collections之间的联系第16章 基类16.1 数学16.1.1 System.Math类16.1.2 System.Random类16.2 时间、日期和持续时间16.2.1 System.DateTime结构16.2.2 System.TimeSpan结构16.2.3 System.Diagnostics.Stopwatch类16.3 驱动器、目录、文件和路径16.3.1 操作驱动器16.3.2 操作目录16.3.3 操作文件16.3.4 操作路径16.3.5 文件系统监视器16.4 注册表16.4.1 引言16.4.2 注册表的结构16.4.3 注册表的层级16.4.4 用.NET读/写注册表16.5 调试16.5.1 用来自定义对象调试时的视图的attribute16.5.2 调试“仅我的代码”16.5.3 调试模式16.5.4 解决调试问题16.6 跟踪16.6.1 监听器16.6.2 跟踪源和源级别16.6.3 筛选跟踪源16.6.4 跟踪缩进16.7 正则表达式16.7.1 引言16.7.2 语法16.7.3 示例16.7.4 .NET和正则表达式16.7.5 优化正则表达式的求值16.8 控制台16.8.1 光标16.8.2 显示16.8.3 大小与位置16.8.4 颜色16.8.5 数据输入16.8.6 重定向控制台的流16.8.7 杂项第17章 输入、输出和流17.1 流的简介17.1.1 流的类层次结构17.1.2 操作流的工具17.1.3 流数据的类型化17.1.4 字符串编码17.2 读写文件17.2.1 简单的文件读写17.2.2 读写二进制数据文件17.2.3 读写文本文件17.2.4 以异步的方式操作流17.3 用套接字来使用TCP/IP17.3.1 套接字和TCP/IP简介17.3.2 以同步的方式操作套接字17.3.3 以异步的方式操作套接字17.4 获取网络接口和状态的信息17.4.1 发现可用网络接口17.4.2 Ping17.4.3 网络发生更改时所触发的事件17.4.4 网络活动统计17.5 HTTP和FTP客户端17.5.1 URI17.5.2 WebClient类17.5.3 通过URI访问资源的其他类17.6 在HTTP.SYS上用HttpListener编HTTP服务器代码17.6.1 HTTP.SYS简介17.6.2 System.Net.HttpListener类17.6.3 以异步方式处理HTTP请求17.7 支持邮件协议17.8 缓冲和压缩数据流17.8.1 通过Decorator设计模式在流上应用服务17.8.2 缓冲流数据17.8.3 压缩数据流17.9 在串行端口上读写数据17.10 支持安全通讯协议：SSL、NTLM和Kerberos17.10.1 安全通讯协议简介17.10.2 安全通讯协议和System.Net.Security命名空间17.10.3 SslStream类17.10.4 NegotiateStream类17.10.5 加密数据流第18章 Windows Forms应用程序18.1 Windows用户界面18.1.1 控制台应用程序与窗口应用程序18.1.2 Windows消息简介18.1.3 Windows应用程序开发的发展18.1.4 System.Windows.Forms命名空间简介18.2 Windows Forms开发简介18.2.1 处理事件18.2.2 编写方法18.2.3 没有Visual Studio的Windows Forms开发18.3 Windows Forms开发工具18.3.1 模态/非模态对话框简介18.3.2 鼠标和键盘事件18.3.3 Paint事件18.3.4 异步处理18.3.5 其他特性18.4 标准控件18.4.1 Windows Forms 2.0控件的层次结构18.4.2 新增控件概述18.5 创建自定义控件18.6 显示和编辑数据18.6.1 Visual Studio 2005提供的工具18.6.2 BindingSource控件18.6.3 用BindingSource来利用

数据源18.6.4 用对象列表作为数据源18.7 Windows Forms和本地化18.8 GDI+18.8.1
System.Drawing.Graphics类18.8.2 System.Drawing.Pen类18.8.3 System.Drawing.Brush类18.8.4 绘制文
本18.8.5 处理图像18.8.6 优化图像的处理18.8.7 动画和双缓冲第19章 ADO.NET 2.019.1 数据库简
介19.1.1 DBMS19.1.2 RDBMS19.1.3 SQL语言19.1.4 分布式架构的必要性19.2 ADO.NET简介19.2.1
连接模式与非连接模式19.2.2 数据提供程序19.2.3 ADO.NET:全局观19.2.4 我们将在范例中使用的DB19.3 连接与数据提供程序19.3.1 解除应用程序与数据提供程序之间的耦合19.3.2 连接字符串19.3.3 如何存储连接字符串以及连接字符串存储在哪里19.3.4 连接池19.3.5 访问数据源的元数据19.4 使用DataReader在连接模式下工作19.4.1 使用DataReader从数据库中获取数据19.4.2 在数据库端进行标量运算19.4.3 使用SQL查询来修改数据19.5 使用DataSet在非连接模式下工作19.5.1 使用来自数据库的数据填充缓存19.5.2 在DataSet中对多表之间的关联进行操作19.5.3 将DataSet中更新的数据保存到数据库19.5.4 非连接模式与乐观/悲观并发策略19.5.5 DataSet中数据表的约束19.5.6
DataView类19.6 强类型DataSet19.6.1 创建强类型DataSet类19.6.2 TableAdapter和强类型SQL请
求19.7 连接模式和非连接模式之间的桥接器19.8 对象与关系数据之间的桥接器19.8.1 结构问
题19.8.2 行为问题19.8.3 解决上述问题的途径19.8.4 用于对象关系映射的.NET工具19.9 SQL Server
数据提供程序的专有功能19.9.1 异步请求19.9.2 批量复制19.9.3 SqlClient连接上的统计数据19.9.4
SQL Server 2005 Express版本第20章 事务20.1 事务简介20.1.1 事务管理器、资源管理器和数据
源20.1.2 分布式事务以及2PC算法简介20.1.3 在SQL Server连接上进行的本地事务20.1.4 分布式事务
协调器20.2 System.Transactions20.2.1 LTM、持久RM和易变RM简介20.2.2 System.Transactions实现的
隐式事务20.2.3 在事务中被触发的事件20.2.4 System.Transactions揭秘20.2.5 事务隔离级别简
介20.2.6 事务作用域20.2.7 System.Transactions实现的显式事务20.3 System.Transactions的高级用
法20.3.1 在多个线程中执行一个事务20.3.2 以异步方式完成事务20.3.3 System.Transactions和CAS20.4
用于实现自定义RM的机制第21章 XML21.1 简介21.1.1 XML所解决的问题21.1.2 统一文档世界
和数据世界21.1.3 XML文档的结构21.2 XSD、XPath、XSLT和XQuery简介21.2.1 用XSD schema类型
化XML文档及其数据21.2.2 XPath21.2.3 XSLT21.2.4 XQuery21.3 遍历和编辑XML文档的方法21.4
使用XmlReader类和XmlWriter类的Cursor方法21.4.1 使用XmlReader类读取数据21.4.2 在读取数据时
对其进行验证21.4.3 使用XmlWriter类编辑数据21.5 使用XmlDocument类的Tree/DOM方法21.5.1 使
用XmlDocument类装载和遍历XML文档21.5.2 使用XmlDocument类编辑和保存XML数据21.5.3 使
用XmlDocument类验证XML文档21.5.4 XmlDocument类的事件21.6 使用XPath遍历和编辑XML文
档21.6.1 对内存中的DOM树应用XPath表达式21.6.2 用XPathNavigator对象遍历XPathDocument对
象21.6.3 用XPathNodeIterator对象遍历XPath的选择结果集21.6.4 用XPathNavigator对象编
辑XmlDocument对象21.7 使用XSLT样式表转换XML文档21.8 连接关系型数据与XML文档的桥接
器21.8.1 从DataSet获取XML文档21.8.2 用XML文档填充DataSet21.8.3 System.Xml.XmlDataDocument
类21.8.4 XML和SQL Server21.9 连接对象与XML文档的桥梁21.9.1 System.Xml.XmlSerialization
类21.9.2 用于XML序列化的attribute21.9.3 sgen.exe工具21.9.4 xsd.exe工具21.10 Visual Studio
和XML21.10.1 创建、查看以及编辑XML文档和XSD schema21.10.2 用XSD schema验证XML文
档21.10.3 编辑和调试XSLT程序第22章 .NET Remoting22.1 简介22.1.1 什么是.NET Remoting22.1.2
FAQ22.2 按引用封送22.3 按值封送和二进制序列化22.4 ObjectHandle类22.5 对象的激活22.5.1
分布式体系的组件22.5.2 宿主概览22.5.3 信道概览22.5.4 同步方式、异步方式和单向方式调用22.5.5
对象激活与对象创建22.6 well-known对象的激活22.7 客户端激活的对象22.7.1 使用new关键字激
活对象22.7.2 潜在的问题22.8 Factory设计模式和soapsuds.exe工具22.8.1 Factory设计模式22.8.2
soapsuds.exe工具22.9 well-known和客户端激活的对象的生命周期22.10 配置.NET Remoting22.10.1
配置宿主22.10.2 配置客户端22.10.3 联合使用接口和配置文件22.11 .NET Remoting服务器的部
署22.11.1 Windows服务22.11.2 IIS22.12 安全的.NET Remoting信道22.12.1 安全的TCP信道22.12.2
安全的HTTP信道22.13 代理和消息22.13.1 把方法调用转换成消息22.13.2 IMessage接口的层次结
构22.13.3 透明代理、真实代理和ObjRef类22.13.4 通过ObjRef类发布对象22.13.5 消息接收器22.13.6
为何考虑自定义真实代理22.13.7 开发自定义真实代理22.13.8 在类的所有实例上使用自定义真实
代理22.13.9 读写方法调用的参数22.14 信道22.14.1 简介22.14.2 发送方信道和代理22.14.3 接收方

信道和服务器对象22.14.4 消息接收器、格式化程序和信道22.14.5 信道接收器提供程序22.14.6 示例
: 显示网络消息的大小22.15 .NET上下文22.15.1 简介22.15.2 上下文绑定和上下文灵活对象22.15.3
上下文attribute和上下文属性22.15.4 消息接收器区域22.15.5 使用区域的示例22.15.6 调用上下
文22.16 小结22.16.1 激活对象的方式22.16.2 截获消息第23章 ASP.NET 2.023.1 简介23.1.1 历
史23.1.2 ASP与ASP.NET23.1.3 ASP.NET 1.x与ASP.NET 2.023.2 ASP.NET概览23.2.1 Web Form23.2.2
运行期的ASP.NET、IIS和Web应用程序23.2.3 在你自己的.NET应用程序中承载ASP.NET23.2.4 基
于HTTP.SYS使用ASP.NET23.3 ASP.NET应用程序的源代码23.3.1 内联代码23.3.2 服务器端脚本的代
码段23.3.3 代码隐藏23.4 编译和部署模型23.4.1 动态编译23.4.2 就地预编译23.4.3 部署预编
译23.5 Web Form和控件23.5.1 服务器控件23.5.2 客户端和服务端之间的交互23.5.3 视图状
态23.5.4 回发事件和非回发事件23.5.5 控件状态23.5.6 跨页面传递23.5.7 HTML服务器控件与Web
服务器控件23.6 页面生命周期23.7 ASP.NET应用程序配置23.7.1 Web.Config文件的组织23.7.2
Web.Config文件的部分23.7.3 processModel部分23.7.4 更新配置23.7.5 在运行期应用配置更新23.8
HTTP管线23.8.1 简介23.8.2 HttpApplication类和Global.asax文件23.8.3 HTTP上下文23.8.4 HTTP
模块23.8.5 HTTP处理程序23.9 状态和会话管理23.9.1 会话管理23.9.2 处理会话标识符23.9.3 会
话存储的标准实现23.9.4 为存储会话提供自定义实现23.10 Provider设计模式23.11 错误处理23.11.1
system.Web 配置元素和 customErrors 配置元素23.11.2 Application_Error事件23.11.3 ErrorPage属
性23.12 跟踪、诊断以及事件管理23.12.1 trace.axd HTTP处理程序23.12.2 ASP.NET性能计数
器23.12.3 ASP.NET状态监视23.13 输入数据的校验23.13.1 校验控件23.13.2 实现自定义校验23.13.3
校验组23.13.4 ValidationSummary类23.14 用户控件23.14.1 复合用户控件23.14.2 用户控件事
件23.14.3 用户控件状态23.14.4 用户控件与Visual Studio的设计时支持23.15 缓存23.15.1 页面缓
存23.15.2 缓存页面的多个版本23.15.3 页面片段缓存23.15.4 缓存后替换23.15.5 数据缓存23.15.6
缓存依赖23.15.7 SQL Server缓存依赖23.15.8 自定义缓存依赖23.16 数据源23.16.1 通过程序的方式
绑定控件和数据源23.16.2 声明式绑定控件和数据源23.16.3 平铺数据源与层次数据源23.16.4
ObjectDataSource类23.16.5 利用数据源更新数据23.17 查看并编辑数据 23.17.1 GridView控
件23.17.2 模板23.17.3 DetailsView控件23.17.4 FormView控件23.17.5 显示XML数据23.18 母版
页23.18.1 母版页和内容页面23.18.2 嵌套母版页23.18.3 配置母版页23.18.4 从内容页面访问母版
页23.19 ASP.NET 2.0与本地化23.20 站点导航23.21 安全23.21.1 通过IIS验证Windows用户23.21.2
ASP.NET验证23.21.3 表单验证提供程序23.21.4 管理用户23.21.5 管理角色23.21.6 安全服务器控
件23.22 个性化与用户配置23.22.1 配置提供程序以及用户数据的管理23.22.2 匿名识别23.22.3 个
性化与会话23.23 样式、主题与皮肤23.23.1 CSS样式和控件23.23.2 主题23.23.3 皮肤23.23.4 命名
的皮肤23.24 WebPart23.24.1 创建包含WebPart的页面23.24.2 设计方式23.24.3 WebPart目录23.24.4
编辑方式23.24.5 连接WebPart第24章 使用.NET进行Web服务开发24.1 简介24.1.1 SOA:面向服务
架构24.1.2 SOAP和WSDL24.1.3 WS-I基本概要24.1.4 消息交换模式24.2 开发一个简单的Web服
务24.2.1 不使用Visual Studio开发一个简单的Web服务24.2.2 使用Visual Studio开发一个简单的Web服
务24.3 测试与调试Web服务24.3.1 测试Web服务24.3.2 调试Web服务24.4 创建Web服务的.NET客
户端24.4.1 不使用Visual Studio来创建Web服务的.NET客户端24.4.2 使用Visual Studio创建Web服
务的.NET客户端24.5 异步调用与消息交换模式24.6 通过.NET Remoting客户端使用Web服务24.7
SOAP消息24.7.1 简介24.7.2 定义以及处理SOAP首部24.7.3 对SOAP消息体编码24.7.4 SOAP错误
消息24.7.5 SOAP与底层传输协议24.8 Web服务契约与WSDL语言24.8.1 WSDL能够表达什么24.8.2
剖析WSDL文档24.9 WSE与WS-*规范简介24.9.1 WSE简介24.9.2 WSE 3.0所支持的规范24.9.3 安
装WSE24.9.4 WSE是如何利用SOAP扩展的24.9.5 使用WSE诊断的第一个测试24.10 WSE尚未支持
的WS-*规范24.10.1 WS-PolicyAttachment与WS-MetadataExchange24.10.2 WS-ReliableMessage24.10.3
UDDI and WS-Discovery24.10.4 WS-Federation24.10.5 WS-Coordination24.10.6
WS-AtomicTransaction与WS-BusinessActivity24.10.7 WS-Enumeration24.10.8 WS-Eventing24.10.9
WS-Management24.11 WCF简介附录A C# 2.0的关键字附录B .NET 2.0的新增功能附录C 设计模
式简介附录D 针对.NET 2.0平台的工具

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>