

<<C#程序设计语言>>

图书基本信息

书名：<<C#程序设计语言>>

13位ISBN编号：9787111282617

10位ISBN编号：7111282612

出版时间：2010-1

出版时间：机械工业出版社

作者：Anders Hejlsberg,Mads Torgersen,Scott Wiltamuth

页数：548

译者：顾雁宏,徐旭铭

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#程序设计语言>>

前言

自2000年夏, NET发布以来已经有8个年头了。

对我来说,当时, NET最重要的两点就是结合了托管代码的本地执行和用于程序之间通信的XML消息机制。

不过那个时候我还没有意识到C#变得那么重要。

C#从一开始就是程序员们理解和使用。

NET的主要手段。

如果你问一个普通的。

NET程序员一个值类型和一个引用类型的区别是什么,通常的回答都是“结构和类的区别”,而非“是否是从System。

Value Type继承而来的类型”。

为什么?因为我们都是用语言,而不是通过API来和运行时(更重要的是,其他人)交流想法和意图的。

如果没有一门出色的语言,一个平台要想成功是不可想象的。

C#最初就为人们如何看待。

NET打下了坚实的基础。

随着。

NET的不断发展,C#的重要性也与日俱增,诸如迭代器和真正的闭包(也叫匿名方法)都是用C#译器实现的纯语言特性,而不是平台特性。

C#3.0的发布更意味着C#为了。

NET不断创新的主角,它引入了标准化的查询操作符,简洁的lambda表达式。

扩展方法,以及在运行时访问表达式树的能力——这些都是通过语言和编译器实现的。

说到C#就一定会提到它的缔造者Anders Hejlsberg。

我非常荣幸地在C#3.0设计阶段连续好几个月参与了C#的设计会议,Anders的工作让我大开眼界。

他那种深谙程序员喜欢什么和不喜欢什么的天赋实在是一流——同时他又能和设计团队紧密合作,并最终获得最佳的设计方案。

特别是在C#3.0上,Anders在从函数式语言社区获取灵感并将它们带给广大群众的过程中层现出无与伦比的能力。

要知道这绝对不是一件容易的事情。

Guy Steele曾经在谈论Java时说到:“我们没打算要吸引Lisp程序员,我们的目标是C++程序员。

我们成功地把他们从转向Lisp的路上吸引了过来。

”当我看到C#3.0的时候,我就知道C#已经至少获得了一名C++程序员(就是我自己)的青睐。

即使C#很出色,但是人们还是需要一份用自然语言(这里是英文)和一些范式(BNF)写成的文档来帮助他们抓住要点,以及任何消除晦涩的地方。

而你手中的这本书正是这样的一份文档。

据我的经验,我敢说每个。

NET程序员在读这本书的时候都至少会有一次“啊,原来如此”的感叹,它能让你的水平更上一层楼。

享受它吧。

<<C#程序设计语言>>

内容概要

C#语言结合了快速应用开发语言的高效和C/C++语言的强大。

现在C# 3.0又加入了函数式编程技术和语言集成查询（LINQ，Language INtegrated Query）。

《C#程序设计语言(原书第3版)》正是C# 3.0的权威技术指南。

这一版由C#的缔造者Anders Hejlsberg和他的同事们合著，全部内容都更新到了C# 3.0版。

《C#程序设计语言(原书第3版)》提供了C# 3.0语言完整的规格说明、参考资料、范例代码和来自九位卓越的C#大师的详细注解。

这些注解所达到的深度和广度是很难在其他书中找到的。

《C#程序设计语言(原书第3版)》的正文介绍了C#的概念，而这些恰到好处的注解则解释了为什么这些特性是重要的，应该怎么使用它们，它们和其他语言的关系是什么，甚至它们是如何进化而来的。

对任何希望深入理解C#的程序员来说，这本书都是不容错过的参考经典。

<<C#程序设计语言>>

作者简介

作者：(美国)海杰尔斯伯格(Anders Hejlsberg) (美国)Mads Torgersen (美国)Scott Wiltamuth 等 译者：顾雁宏 徐旭铭 Ander-Hejlsberg是编程界的传奇人物。

他是C#语言的架构师，同时也是微软技术专家。

他在1996年加入微软，之前13年的职业生涯则是在Borland度过，他曾经是Delphi和Turbo Pascal的首席架构师。

Mads Torgersen是微软的资深程序经理。

作为C#程序经理，他负责召开C#语言的设计会议及维护C#语言的规范。

在2005年加入微软之前，Mads是奥尔胡斯大学的副教授，主要教授和研究面向对象编程语言。

在那里，他领导的小组设计实现了Java的泛型通配符。

Scott Wiltamuth是Visual Studio的合作程序经理。

他在微软做过很多面向程序员的项目，包括Visual Basic、VBScript、Jscript、Visual J++和Visual C#。

Scott是C#语言的设计师之一，他拥有斯坦福大学计算机科学硕士学位。

Peter Golde在离开微软之前是微软C#编译器的首席程序员。

他作为微软在ECMA委员会（这个委员会负责了c#的标准化工作）的主要代表，领导实现了编译器并参与了语言的设计。

书籍目录

序 作者简介 注解者简介 前言 第1章 介绍 1.1 Hello,World 1.2 程序结构 1.3 类型和变量 1.4 表达式 1.5 语句 1.6 类和对象 1.6.1 成员 1.6.2 访问控制 1.6.3 类型参数 1.6.4 基类 1.6.5 字段 1.6.6 方法 1.6.7 其他函数成员 1.7 结构 1.8 数组 1.9 接口 1.10 枚举 1.11 委托 1.12 特性 第2章 词法结构 2.1 程序 2.2 文法 2.2.1 文法表示法 2.2.2 词法文法 2.2.3 语法文法 2.3 词法分析 2.3.1 行终结符 2.3.2 注释 2.3.3 空白符 2.4 标记 2.4.1 Unicode字符转义序列 2.4.2 标识符 2.4.3 关键字 2.4.4 字量 2.4.5 操作符和标点符号 2.5 预处理指令 2.5.1 条件编译符号 2.5.2 预处理表达式 2.5.3 声明指令 2.5.4 条件编译指令 2.5.5 诊断指令 2.5.6 区域指令 2.5.7 行指令 2.5.8 编译指示指令 第3章 基本概念 3.1 应用程序起始 3.2 应用程序终止 3.3 声明 3.4 成员 3.4.1 命名空间成员 3.4.2 结构成员 3.4.3 枚举成员 3.4.4 类成员 3.4.5 接口成员 3.4.6 数组成员 3.4.7 委托成员 3.5 成员访问 3.5.1 声明可访问性 3.5.2 可访问域 3.5.3 实例成员的保护访问 3.5.4 访问限制 3.6 签名和重载 3.7 作用域 3.7.1 名字隐藏 3.8 命名空间和类型名称 3.8.1 完全限定名 3.9 自动化内存管理 3.10 执行顺序 第4章 类型 4.1 值类型 4.1.1 System.ValueType类型 4.1.2 默认构造函数 4.1.3 结构类型 4.1.4 简单类型 4.1.5 整数类型 4.1.6 浮点数类型 4.1.7 decimal类型 4.1.8 bool类型 4.1.9 枚举类型 4.1.10 可空值类型 4.2 引用类型 4.2.1 类类型 4.2.2 Object类型 4.2.3 String类型 4.2.4 接口类型 4.2.5 数组类型 4.2.6 委托类型 4.3 装箱和拆箱 4.3.1 装箱转换 4.3.2 拆箱转换 4.4 构造类型 4.4.1 类型实参 4.4.2 开放式和封闭式类型 4.4.3 绑定和未绑定类型 4.4.4 满足限制 4.5 类型参数 4.6 表达式树类型 第5章 变量 5.1 变量类别 5.1.1 静态变量 5.1.2 实例变量 5.1.3 数组元素 5.1.4 值参数 5.1.5 引用参数 5.1.6 输出参数 5.1.7 局部变量 5.2 默认值 5.3 明确赋值 5.3.1 初始赋值的变量 5.3.2 未赋初值的变量 5.3.3 确定明确赋值的精确规则 5.4 变量引用 5.5 变量引用的原子性 第6章 转换 6.1 隐式转换 6.1.1 标识转换 6.1.2 隐式数字转换 6.1.3 隐式枚举转换 6.1.4 隐式可空值转换 6.1.5 Null字量转换 6.1.6 隐式引用转换 6.1.7 装箱转换 6.1.8 隐式常量表达式转换 6.1.9 带类型参数的隐式转换 6.1.10 自定义隐式转换 6.1.11 匿名函数转换和方法组转换 6.2 显式转换 6.2.1 显式数字转换 6.2.2 显式枚举转换 6.2.3 显式可空值转换 6.2.4 显式引用转换 6.2.5 拆箱转换 6.2.6 带类型参数的显式转换 6.2.7 自定义显式转换 6.3 标准转换 6.3.1 标准隐式转换 6.3.2 标准显式转换 6.4 自定义转换 6.4.1 允许的自定义转换 6.4.2 提升转换操作符 6.4.3 自定义转换的计算 6.4.4 自定义隐式转换 6.4.5 自定义显式转换 6.5 匿名函数转换 6.5.1 匿名函数到委托类型转换的计算 6.5.2 匿名函数到表达式树类型转换的计算 6.5.3 实现举例 6.6 方法组转换 第7章 表达式 7.1 表达式分类 7.1.1 表达式的值 7.2 操作符 7.2.1 操作符优先级和结合性 7.2.2 操作符重载 7.2.3 一元操作符重载决策 7.2.4 二元操作符重载决策 7.2.5 候选自定义操作符 7.2.6 数字提升 7.2.7 提升操作符 7.3 成员查找 7.3.1 基础类型 7.4 函数成员 7.4.1 参数列表 7.4.2 类型推导 7.4.3 重载决策 7.4.4 函数成员调用 7.5 基础表达式 7.5.1 字量 7.5.2 简单名字 7.5.3 括号表达式 7.5.4 成员访问 7.5.5 调用表达式 7.5.6 元素访问 7.5.7 this访问 7.5.8 base访问 7.5.9 后缀递增和递减操作符 7.5.10 new操作符 7.5.11 typeof操作符 7.5.12 checked和unchecked操作符 7.5.13 默认值表达式 7.5.14 匿名方法表达式 7.6 一元操作符 7.6.1 一元加号操作符 7.6.2 一元减号操作符 7.6.3 逻辑否操作符 7.6.4 按位求补操作符 7.6.5 前缀递增和递减操作符 7.6.6 转换表达式 7.7 算术操作符 7.7.1 乘法操作符 7.7.2 除法操作符 7.7.3 求余操作符 7.7.4 加法操作符 7.7.5 减法操作符 7.8 移位操作符 7.9 关系和类型测试操作符 7.9.1 整数比较操作符 7.9.2 浮点数比较操作符 7.9.3 小数比较操作符 7.9.4 布尔值相等操作符 7.9.5 枚举比较操作符 7.9.6 引用类型相等操作符 7.9.7 字符串相等操作符 7.9.8 委托相等操作符 7.9.9 相等操作符和null 7.9.10 is操作符 7.9.11 as操作符 7.10 逻辑操作符 7.10.1 整数逻辑操作符 7.10.2 枚举逻辑操作符 7.10.3 布尔值逻辑操作符 7.10.4 可空值布尔逻辑操作符 7.11 条件逻辑操作符 7.11.1 布尔条件逻辑操作符 7.11.2 自定义条件逻辑操作符 7.12 Null拼接操作符 7.13 条件操作符 7.14 匿名函数表达式 7.14.1 匿名函数签名 7.14.2 匿名函数主体 7.14.3 重载决策 7.14.4 外部变量 7.14.5 匿名函数表达式的计算 7.15 查询表达式 7.15.1 查询表达式里的歧义 7.15.2 查询表达式翻译 7.15.3 查询表达式模式 7.16 值操作符 7.16.1 简单赋值 7.16.2 组合赋值 7.16.3 事件赋值 7.17 达式 7.18 量表达式 7.19 尔表达式 第8章 语句 8.1 终点和可及性 8.2 块 8.2.1 语句列表 8.3 空语句 8.4 标签语句 8.5 声明语句 8.5.1 局部变量声明 8.5.2 局部常量声明 8.6 表达式语句 8.7 选择语句 8.7.1 if语句 8.7.2 switch语句 8.8 迭代语句 8.8.1 while语句 8.8.2 do语句 8.8.3 for语句 8.8.4 for each语句 8.9 跳转语句 8.9.1 break语句 8.9.2 continue语句 8.9.3 go to语句 8.9.4 return语句 8.9.5 throw语句 8.10 try语句 8.11 checked和unchecked语句 8.12 lock语句 8.13 using语句 8.14 yield语句 第9章 命名空间 9.1 编译单元 9.2 命名空间声明 9.3 Extern别名 9.4 using指令 9.4.1 using别名指令 9.4.2 using命名空间指令 9.5 命名空间成员 9.6 类型声明 9.7 命名空间别名限定

符9.7.1 别名的唯一性第10章 类10.1 类声明10.1.1 类修饰符10.1.2 partial修饰符10.1.3 类型形参10.1.4 类基础规范10.1.5 类型形参限制10.1.6 类主体10.2 局部类型10.2.1 特性10.2.2 修饰符10.2.3 类型形参和限制10.2.4 基类10.2.5 基础接口10.2.6 成员10.2.7 局部方法10.2.8 名字绑定10.3 类成员10.3.1 实例类型10.3.2 构造类型的成员10.3.3 继承10.3.4 new修饰符10.3.5 访问修饰符10.3.6 组成类型10.3.7 静态成员和实例成员10.3.8 嵌套类型10.3.9 保留成员名10.4 常量10.5 字段10.5.1 静态字段和实例字段10.5.2 只读字段10.5.3 易失字段10.5.4 字段初始化10.5.5 字段初始化语句10.6 方法10.6.1 方法形参10.6.2 静态和实例方法10.6.3 虚拟方法10.6.4 覆写方法10.6.5 密封方法10.6.6 抽象方法10.6.7 外部方法10.6.8 局部方法10.6.9 扩展方法10.6.10 方法主体10.6.11 方法重载10.7 属性10.7.1 静态属性和实例属性10.7.2 访问器10.7.3 自动实现的属性10.7.4 可访问性10.7.5 虚拟.密封.覆写和抽象访问器10.8 事件10.8.1 类似字段的事件10.8.2 事件访问器10.8.3 静态事件和实例事件10.8.4 虚拟.密封.覆写和抽象访问器10.9 索引10.9.1 索引重载10.10 操作符10.10.1 一元操作符10.10.2 二元操作符10.10.3 转换操作符10.11 实例构造函数10.11.1 构造函数初始化语句10.11.2 实例字段初始化语句10.11.3 构造函数的执行10.11.4 默认构造函数10.11.5 私有构造函数10.11.6 可选的实例构造函数参数10.12 静态构造函数10.13 析构函数10.14 迭代器10.14.1 计数接口10.14.2 枚举接口10.14.3 Yield类型10.14.4 计数对象10.14.5 枚举对象10.14.6 实现举例第11章 结构11.1 结构声明11.1.1 结构修饰符11.1.2 partial修饰符11.1.3 结构接口11.1.4 结构主体11.2 结构成员11.3 类和结构的区别11.3.1 值语义11.3.2 继承11.3.3 赋值11.3.4 默认值11.3.5 装箱和拆箱11.3.6 this的含义11.3.7 字段初始化语句11.3.8 构造函数11.3.9 析构函数11.3.10 静态构造函数11.4 结构举例11.4.1 数据库整数类型11.4.2 数据库布尔类型第12章 数组12.1 数组类型12.1.1 System. Array类型12.1.2 数组和泛型I List接口12.2 数组创建12.3 数组元素访问12.4 数组成员12.5 数组协变12.6 数组初始化语句第13章 接口13.1 接口声明13.1.1 接口修饰符13.1.2 partial修饰符13.1.3 基础接口13.1.4 接口主体13.2 接口成员13.2.1 接口方法13.2.2 接口属性13.2.3 接口事件13.2.4 接口索引13.2.5 接口成员访问13.3 完全限定接口成员名13.4 接口实现13.4.1 显式接口成员实现13.4.2 实现接口的唯一性13.4.3 泛型方法的实现13.4.4 接口映射13.4.5 接口实现继承13.4.6 重新实现接口13.4.7 抽象类和接口第14章 枚举14.1 枚举声明14.2 枚举修饰符14.3 枚举成员14.4 System. Enum类型14.5 枚举值和操作第15章 委托15.1 委托声明15.2 委托兼容性15.3 委托实例化15.4 委托调用第16章 异常16.1 异常产生的原因16.2 System. Exception类16.3 异常是如何处理的16.4 常见的异常类第17章 特性17.1 特性类17.1.1 特性的用法17.1.2 位置和已命名参数17.1.3 特性形参类型17.2 特性规范17.3 特性实例17.3.1 特性的编译17.3.2 在运行时获取一个特性实例17.4 保留特性17.4.1 Attribute Usage特性17.4.2 Conditional特性17.4.3 Obsolete特性17.5 用于互操作的特性17.5.1 和COM以及Win32组件互操作17.5.2 和其他.NET语言互操作第18章 不安全的代码18.1 不安全的上下文18.2 指针类型18.3 固定变量和可移动变量18.4 指针转换18.4.1 指针数组18.5 表达式里的指针18.5.1 指针间接寻址18.5.2 指针成员访问18.5.3 指针元素访问18.5.4 取地址操作符18.5.5 指针递增和递减18.5.6 指针算术18.5.7 指针比较18.5.8 size of操作符18.6 fixed语句18.7 定长缓冲区18.7.1 定长缓冲区声明18.7.2 表达式里的定长缓冲区18.7.3 明确赋值检查18.8 栈分配18.9 动态内存分配附录A 文档注释附录B 文法附录C 参考资料

章节摘录

插图：5.3.2 未赋初值的变量下列的变量类别都被归类为还未赋初值的：
· 结构变量里还未赋初值的实例变量。

- 输出参数，包括结构实例构造函数里的this变量。
- 局部变量，除了那些在catch子句或for each语句里声明的以外。

5.3.3 确定明确赋值的精确规则编译器必须采用一套和在这一节里的描述等价的流程来确定每个使用的变量都是明确赋值的。

编译器会处理每一个拥有一到多个还未赋初值变量的函数成员的主体。

对于每一个还未赋初值的变量v，编译器会在函数成员里以下的每一个地方确认v的明确赋值状态：
· 在每个语句的开始。

- 在每个语句的终点（8.1节）。
- 在每个arc上控制转移到另一个语句或者到语句的终点。
- 在每个表达式的开始。
- 在每个表达式的终点。

v的明确赋值状态可以是：
· 明确赋值。

这表示在通向这一点的所有可能的控制流上，v都已经被赋值。

- 未明确赋值。

对于在bool类型的表达式结尾的变量状态，还未明确赋值的变量状态可以（但不一定）属于以下的子状态之一：
· 在true表达式之后明确赋值。

这个状态表示：如果布尔表达式的求值结果为true，那么v就是明确赋值的；但要是它的求值结果为false，那么就不一定会被赋值。

在false表达式之后明确赋值。

这个状态表示：如果布尔表达式的求值结果为false，那么v就是明确赋值的；但要是它的求值结果为true，那么就不一定会被赋值。

下面的规则控制了变量v的状态是如何在每一个位置上被决定的。

5.3.3.1 语句的一般规则 · v在函数成员主体的开始是还未明确赋值的。

- v在任何不可及的语句开始都是明确赋值的。

<<C#程序设计语言>>

媒体关注与评论

“据我的经验，我敢说每个程序员在读这本书的时候都至少会有一次‘啊，原来如此’的感叹，它能让你的编程水平更上一层楼。

” —— 选自Don Box的序

<<C#程序设计语言>>

编辑推荐

《C#程序设计语言 (原书第3版)》：开发人员专业技术丛书

<<C#程序设计语言>>

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>