

你必须知道的.NET



[你必须知道的.NET 下载链接1](#)

著者:王涛

出版者:电子工业出版社

出版时间:2011-8

装帧:

isbn:9787121141287

由王涛编著的《你必须知道的.NET》来自于微软MVP的最新技术心得和感悟，将技术问题以生动易懂的语言展开，层层深入，以例说理。全书主要

包括了.NET基础知识及其深度分析，以.NET Framework和CLR研究为核心展开.NET本质论述，涵盖了.NET基本知识几乎所有的重点内容。全书分为5个部分，第1部分讲述.NET与面向对象，从底层实现角度分析了.NET如何实现面向对象机制，进一步分析了面向对象设计原则；第2部分论述了.NET类型系统和CLR的内存管理机制，并对IL语言进行了相应介绍；第3部分论述.NET Framework框架的方方面面，详细分析了.NET框架的所有重点、难点和疑点内容，对框架类库的全貌进行了必要的专题性探讨；第4部分重点介绍了.NET泛型和安全性的相关知识和本质解密；第5部分对.NET 3.0/3.5/4.0新特性进行了详细的介绍和引导，对于快速入门.NET新特性提供了方便之门。

《你必须知道的.NET》适于对.NET有一定了解的技术学习者、软件工程师和系统架构师阅读，同时也有助于.NET初学者进行快速提高，可作为大中专院校和.NET技术培训机构的参考教材。

作者介绍：

王涛，网名anytao，软件架构师，机械工程硕士，连续三届Microsoft Visual C# MVP，博客园技术专家，著有《你必须知道的.NET》一书，专注于.NET底层架构和.NET平台企业级应用，长期投入于互联网产品开发、微软云计算平台、分布式系统和企业级系统架构的研究与实践。目前，投身于移动互联产业，和几个志同道合的兄弟在技术之路上狂奔。

作者对.NET基础架构和CLR底层运行机制有浓厚的研究兴趣和造诣，熟悉ASP.NET、Windows Azure、SQL Server、WCF、LINQ、Silverlight、IIS、XML、Windows Phone、Facebook相关技术，精通数据库应用系统和大型Web系统的开发流程、技术体系和架构设计，对面向对象、设计模式和软件架构有长期的研究与实践经验。

目录: 第1部分 源流——.NET与面向对象
 第1章 OO大智慧 2
 1.1 对象的旅行 3
 1.1.1 引言 3
 1.1.2 出生 3
 1.1.3 旅程 3
 1.1.4 插曲 4
 1.1.5 消亡 6
 1.1.6 结论 7
 1.2 什么是继承 7
 1.2.1 引言 7
 1.2.2 基础为上 7
 1.2.3 继承本质论 9
 1.2.4 秘境追踪 13
 1.2.5 规则制胜 16
 1.2.6 结论 17
 1.3 封装的秘密 17
 1.3.1 引言 17
 1.3.2 让ATM告诉你，什么是封装 17
 1.3.3 秘密何处：字段、属性和方法 19
 1.3.4 封装的意义 23
 1.3.5 封装规则 23
 1.3.6 结论 24
 1.4 多态的艺术 24
 1.4.1 引言 24
 1.4.2 问题的抛出 24
 1.4.3 最初的实现 25
 1.4.4 多态，救命的稻草 27
 1.4.5 随需而变的业务 30
 1.4.6 多态的类型、本质和规则 31
 1.4.7 结论 33
 1.5 玩转接口 34
 1.5.1 引言 34
 1.5.2 什么是接口 34
 1.5.3 .NET中的接口 35
 1.5.4 面向接口的编程 38
 1.5.5 接口之规则 40
 1.5.6 结论 40
 1.5.7 参考文献 40
 第2章 OO大原则 41
 2.1 OO原则综述 42
 2.1.1 引言 42
 2.1.2 讲述之前 42
 2.1.3 原则综述 43
 2.1.4 学习建议 44
 2.1.5 结论 44
 2.2 单一职责原则 44
 2.2.1 引言 44
 2.2.2 引经据典 45
 2.2.3 应用反思 45
 2.2.4 规则建议 47
 2.2.5 结论 48
 2.3 开放封闭原则 48
 2.3.1 引言 48
 2.3.2 引经据典 48
 2.3.3 应用反思 49
 2.3.4 规则建议 52
 2.3.5 结论 53
 2.4 依赖倒置原则 53
 2.4.1 引言 53

引经据典 53 2.4.3 应用反思 54 2.4.4 规则建议 56 2.4.5 结论 56 2.5 接口隔离原则 56 2.5.1
引言 56 2.5.2 引经据典 56 2.5.3 应用反思 57 2.5.4 规则建议 59 2.5.5 结论 59 2.6
Liskov 替换原则 59 2.6.1 引言 59 2.6.2 引经据典 59 2.6.3 应用反思 60 2.6.4 规则建议 61
2.6.5 结论 62 参考文献 62 第3章 OO之美 63 3.1 设计的分寸 64 3.1.1 引言 64 3.1.2
设计由何而来 64 3.1.3 从此重构 65 3.1.4 结论 67 3.2 依赖的哲学 67 3.2.1 引言 67 3.2.2
什么是依赖，什么是抽象 68 3.2.3 重新回到依赖倒置 73 3.2.4
解构控制反转 (IoC) 和依赖注入 3.2.4 (DI) 79 3.2.5 典型的设计模式 82 3.2.6
基于契约编程：SOA 架构下的 3.2.6 依赖 83 3.2.7 对象创建的依赖 84 3.2.8 不规则总结 87
3.2.9 结论 87 3.3 模式的起点 87 3.3.1 引言 87 3.3.2 模式的起点 88 3.3.3 模式的建议 90
3.3.4 结论 91 13.5.5 结论 466 13.6 江湖一统：WPF、WCF、WF 467 13.6.1 引言 467
13.6.2 WPF 467 13.6.3 WCF 468 13.6.4 WF 469 13.6.5 结论 470 参考文献 470 第14章
跟随.NET 4.0 脚步 472 14.1 .NET 十年 473 14.1.1 引言 473 14.1.2 历史脚步 473 14.1.3
未来之变 477 14.1.4 结论 479 14.2 .NET 4.0，第一眼 480 14.2.1 引言 480 14.2.2 第一眼
481 14.2.3 结论 484 14.3 动态变革：dynamic 484 14.3.1 引言 484 14.3.2 初探 485 14.3.3
本质：DLR 485 14.3.4 PK 解惑 488 14.3.5 应用：动态编程 490 14.3.6 结论 491 14.4
趋势必行，并行计算 491 14.4.1 引言 491 14.4.2 拥抱并行 492 14.4.3 TPL 493 14.4.4
PLINQ 495 14.4.5 并行补遗 496 14.4.6 结论 497 14.5 命名参数和可选参数 497 14.5.1
引言 497 14.5.2 一览究竟 498 14.5.3 简单应用 499 14.5.4 结论 499 14.6 协变与逆变 500
14.6.1 引言 500 14.6.2 概念解析 500 14.6.3 深入 502 14.6.4 结论 504 14.7 Lazy<T> 点滴
504 14.7.1 引言 505 14.7.2 延迟加载 505 14.7.3 Lazy<T> 登场 505 14.7.4 Lazy<T> 本质
507 14.7.5 结论 509 14.8 Tuple 509 14.8.1 引言 509 14.8.2 Tuple 为何物 510 14.8.3
Tuple Inside 511 14.8.4 优略之间 513 14.8.5 结论 514 参考文献
514 后记：我写的不是代码 516 编后记：遇见幸福 521 3.4 面向对象和基于对象 91 3.4.1
引言 91 3.4.2 基于对象 91 3.4.3 二者的差别 91 3.4.4 结论 92 3.5 也谈.NET 闭包 92 3.5.1
引言 92 3.5.2 什么是闭包 92 3.5.3 .NET 也有闭包 93 3.5.4 福利与问题 95 3.5.5 结论 96 3.6
好代码和坏代码 96 3.6.1 引言 96 3.6.2 好代码、坏代码 97 3.6.3 结论 105 参考文献
105 第2部分 本质——.NET 深入浅出第4章 一切从IL开始 108 4.1
从Hello, world 开始认识IL 109 4.1.1 引言 109 4.1.2 从Hello, world 开始 109 4.1.3
IL 体验中心 109 4.1.4 结论 113 4.2 教你认识IL 代码——从基础到工具 113 4.2.1 引言 113
4.2.2 使用工具 113 4.2.3 为何而探索 115 4.2.4 结论 116 4.3
教你认识IL 代码——IL 语言基础 116 4.3.1 引言 116 4.3.2 变量的声明 116 4.3.3 基本类型
117 4.3.4 基本运算 118 4.3.5 数据加载与保存 118 4.3.6 流程控制 119 4.3.7 结论 120 4.4
管窥元数据和IL 120 4.4.1 引言 120 4.4.2 初次接触 120 4.4.3 继续深入 123 4.4.4
元数据是什么 125 4.4.5 IL 是什么 128 4.4.6 元数据和IL 在JIT 编译时 130 4.4.7 结论 134 4.5
经典指令解析之实例创建 134 4.5.1 引言 134 4.5.2 newobj 和 initobj 134 4.5.3 ldstr 136
4.5.4 newarr 137 4.5.5 结论 139 4.6 经典指令解析之方法调度 140 4.6.1 引言 140 4.6.2
方法调度简论：call、callvirt 和 4.6.2 calli 140 4.6.3 直接调度 142 4.6.4 间接调度 146
4.6.5 动态调度 147 4.6.6 结论 147 参考文献 147 第5章 品味类型 148 5.1
品味类型——从通用类型系统开始 149 5.1.1 引言 149 5.1.2 基本概念 149 5.1.3
位置与关系 150 5.1.4 通用规则 151 5.1.5 结论 152 5.2 品味类型——值类型与引用类型
152 5.2.1 引言 152 5.2.2 内存有理 152 5.2.3 通用规则与比较 156 5.2.4
对症下药——应用场合与注意 5.2.4 事项 158 5.2.5 再论类型判等 159 5.2.6 再论类型转换
159 5.2.7 以代码剖析 160 5.2.8 结论 167 5.3 参数之惑——传递的艺术 167 5.3.1 引言 168
5.3.2 参数基础论 168 5.3.3 传递的基础 169 5.3.4 深入讨论，传递的艺术 170 5.3.5 结论
174 5.4 皆有可能——装箱与拆箱 175 5.4.1 引言 175 5.4.2 品读概念 176 5.4.3 原理分拆
176 5.4.4 还是性能 179 5.4.5 重在应用 180 5.4.6 结论 182 参考文献 182 第6章 内存天下
184 6.1 内存管理概要 185 6.1.1 引言 185 6.1.2 内存管理概观要论 185 6.1.3 结论 186 6.2
对象创建始末 186 6.2.1 引言 187 6.2.2 内存分配 187 6.2.3 结论 193 6.3 垃圾回收 193
6.3.1 引言 193 6.3.2 垃圾回收 193 6.3.3 非托管资源清理 197 6.3.4 结论 204 6.4
性能优化的多方探讨 204 6.4.1 引言 204 6.4.2 性能条款 204 6.4.3 结论 210 参考文献
211 第3部分 格局——.NET 面面俱到第7章 深入浅出——关键字的秘密 214 7.1 把 new 说透
215 7.1.1 引言 215 7.1.2 基本概念 215 7.1.3 深入浅出 217 7.1.4 结论 219 7.2 base 和 this
219 7.2.1 引言 219 7.2.2 基本概念 219 7.2.3 深入浅出 220 7.2.4 通用规则 224 7.2.5 结论
224 7.3 using 的多重身份 224 7.3.1 引言 224 7.3.2 引入命名空间 225 7.3.3 创建别名 225

7.3.4 强制资源清理 227 7.3.5 结论 230 7.4 认识全面的null 230 7.4.1 引言 230 7.4.2
从什么是null开始 230 7.4.3 Nullable<T> (可空类型) 232 7.4.4 ??运算符 234 7.4.5
Null Object模式 235 7.4.6 结论 238 7.5 转换关键字 238 7.5.1 引言 239 7.5.2
自定义类型转换探讨 239 7.5.3 本质分析 240 7.5.4 结论 242 7.6 预处理指令关键字 242
7.6.1 引言 242 7.6.2 预处理指令简述 242 7.6.3 #if、#else、#elif、#endif 243 7.6.4
#define、#undef 244 7.6.5 #warning、#error 244 7.6.6 #line 245 7.6.7 结论 245 7.7
非主流关键字 245 7.7.1 引言 245 7.7.2 checked/unchecked 246 7.7.3 yield 247 7.7.4 lock
250 7.7.5 unsafe 252 7.7.6 sealed 253 7.7.7 结论 254 参考文献 254 第8章
巅峰对决——走出误区 255 8.1 什么才是不变：const和readonly 256 8.1.1 引言 256 8.1.2
从基础到本质 257 8.1.3 比较，还是规则 259 8.1.4 进一步的探讨 260 8.1.5 结论 263 8.2
后来居上：class和struct 263 8.2.1 引言 263 8.2.2 基本概念 263 8.2.3 相同点和不同点
264 8.2.4 经典示例 265 8.2.5 结论 268 8.3 历史纠葛：特性和属性 268 8.3.1 引言 268 8.3.2
概念引入 268 8.3.3 通用规则 270 8.3.4 特性的应用 271 8.3.5 示例 273 8.3.6 结论 277 8.4
面向抽象编程：接口和抽象类 277 8.4.1 引言 277 8.4.2 概念引入 277 8.4.3
相同点和不同点 279 8.4.4 经典示例 281 8.4.5 他山之石 283 8.4.6 结论 283 8.5
恩怨情仇：is和as 284 8.5.1 引言 284 8.5.2 概念引入 284 8.5.3 原理与示例说明 284 8.5.4
结论 285 8.6 貌合神离：覆写和重载 286 8.6.1 引言 286 8.6.2 认识覆写和重载 286 8.6.3
在多态中的应用 288 8.6.4 比较，还是规则 289 8.6.5 进一步的探讨 290 8.6.6 结论 292 8.7
有深有浅的克隆：浅拷贝和深拷贝 292 8.7.1 引言 292 8.7.2 从对象克隆说起 292 8.7.3
浅拷贝和深拷贝的实现 294 8.7.4 结论 296 8.8 动静之间：静态和非静态 296 8.8.1 引言
296 8.8.2 一言蔽之 297 8.8.3 分而治之 297 8.8.4 结论 302 8.9 集合通论 302 8.9.1 引言 302
8.9.2 中心思想——纵论集合 303 8.9.3 各分秋色——.NET集合类大观 307 8.9.4
自我成全——实现自定义集合 314 8.9.5 结论 317 参考文献 317 第9章
本来面目——框架诠释 318 9.1 万物归宗：System.Object 319 9.1.1 引言 319 9.1.2 初识
319 9.1.3 分解 320 9.1.4 插曲：消失的成员 323 9.1.5 意义 325 9.1.6 结论 325 9.2
规则而定：对象判等 325 9.2.1 引言 326 9.2.2 本质分析 326 9.2.3 覆写Equals方法 329
9.2.4 与GetHashCode方法同步 331 9.2.5 规则 332 9.2.6 结论 332 9.3
疑而不惑：interface “继承” 争议 332 9.3.1 引言 332 9.3.2 从面向对象寻找答案 333
9.3.3 以IL探求究竟 334 9.3.4 System.Object真是9.3.4 “万物之宗”吗 334 9.3.5
接口的继承争议 335 9.3.6 结论 335 9.4 给力细节：深入类型构造器 336 9.4.1
引言：一个故事 336 9.4.2 认识对象构造器和类型构造器 337 9.4.3 深入执行过程 339
9.4.4 回归故事 341 9.4.5 结论 342 9.5 如此特殊：大话String 342 9.5.1 引言 342 9.5.2
问题迷局 343 9.5.3 什么是string 345 9.5.4 字符串创建 345 9.5.5 字符串恒定性 346 9.5.6
字符串驻留 (String Interning) 346 9.5.7 字符串操作典籍 350 9.5.8
补充的礼物：StringBuilder 352 9.5.9 结论 354 9.6 简易不简单：认识枚举 354 9.6.1 引言
355 9.6.2 枚举类型解析 355 9.6.3 枚举种种 358 9.6.4 位枚举 360 9.6.5 规则与意义 361
9.6.6 结论 361 9.7 一脉相承：委托、匿名方法和Lambda表达式 362 9.7.1 引言 362
9.7.2 解密委托 362 9.7.3 委托和事件 365 9.7.4 匿名方法 367 9.7.5 Lambda表达式 368
9.7.6 规则 368 9.7.7 结论 369 9.8 Name这回事儿 369 9.8.1 引言 369 9.8.2 畅聊Name 369
9.8.3 回到问题 371 9.8.4 结论 371 9.9 直面异常 371 9.9.1 引言 372 9.9.2 为何而抛 372
9.9.3 从try/catch/finally说起：解析异常 9.9.3 机制 375 9.9.4 .NET系统异常类 377 9.9.5
定义自己的异常类 379 9.9.6 异常法则 381 9.9.7 结论 382 参考文献 382 第10章
格局之选——命名空间剖析 383 10.1 基础——.NET框架概览 384 10.1.1 引言 384 10.1.2
框架概览 384 10.1.3 历史变迁 385 10.1.4 结论 387 10.2 布局——框架类库研究 387 10.2.1
引言 387 10.2.2 为什么了解 388 10.2.3 框架类库的格局 388 10.2.4 一点补充 389 10.2.5
结论 390 10.3 根基——System命名空间 391 10.3.1 引言 391 10.3.2 从基础类型说起 391
10.3.3 基本服务 392 10.3.4 结论 394 10.4 核心——System次级命名空间 394 10.4.1 引言
394 10.4.2 System.IO 395 10.4.3 System.Diagnostics 396 10.4.4
System.Runtime.Serialization 和 10.4.4 System.Xml.Serialization 397 10.4.5 结论 399
参考文献 399 第4部分 拾遗——.NET也有春天第11章 接触泛型 402 11.1 追溯泛型 403
11.1.1 引言 403 11.1.2 推进思维，为什么泛型 403 11.1.3 解析泛型——运行时本质 405
11.1.4 结论 406 11.2 了解泛型 406 11.2.1 引言 406 11.2.2 领略泛型——基础概要 406
11.2.3 典型.NET泛型类 409 11.2.4 基础规则 410 11.2.5 结论 411 11.3 深入泛型 411 11.3.1
引言 411 11.3.2 泛型方法 411 11.3.3 泛型接口 413 11.3.4 泛型委托 415 11.3.5 结论 415

11.4 实践泛型 416 11.4.1 引言 416 11.4.2 最佳实践 416 11.4.3 结论 421 参考文献
421第12章 如此安全性 422 12.1 怎么样才算是安全 423 12.1.1 引言 423 12.1.2
怎么样才算安全 423 12.1.3 .NET安全模型 423 12.1.4 结论 424 12.2 代码访问安全 424
12.2.1 引言 424 12.2.2 证据 (Evidence) 425 12.2.3 权限 (Permission) 和权限集 426
12.2.4 代码组 (Code Group) 428 12.2.5 安全策略 (Security Policy) 428 12.2.6
规则总结 429 12.2.7 结论 430 12.3 基于角色的安全 430 12.3.1 引言 430 12.3.2
Principal (主体) 430 12.3.3 Identity (标识) 431 12.3.4 PrincipalPermission 432
12.3.5 应用示例 432 12.3.6 结论 433 参考文献 433第5部分 未来—.NET技术展望第13章
走向.NET 3.0/3.5变革 436 13.1 品读新特性 437 13.1.1 引言 437 13.1.2 .NET新纪元 437
13.1.3 程序语言新特性 438 13.1.4 WPF、WCF、WF 438 13.1.5 Visual Studio 2008体验
439 13.1.6 其他 439 13.1.7 结论 439 13.2 赏析C# 3.0 439 13.2.1 引言 440 13.2.2
对象初始化器 13.2.2 (Object Initializers) 440 13.2.3 集合初始化器 13.2.3 (Collection
Initializers) 441 13.2.4 自动属性 13.2.4 (Automatic Properties) 442 13.2.5
隐式类型变量 (Implicitly Typed Local Variables) 和 隐式类型数组 13.2.5
(Implicitly Typed Array) 444 13.2.6 匿名类型 (Anonymous Type) 445 13.2.7
扩展方法 13.2.7 (Extension Methods) 446 13.2.8 查询表达式 13.2.8 (Query
Expressions) 448 13.2.9 结论 448 13.3 LINQ体验 449 13.3.1 引言 449 13.3.2 LINQ概览
449 13.3.3 查询操作符 451 13.3.4 LINQ to XML示例 451 13.3.5 规则 453 13.3.6 结论 453
13.4 LINQ江湖 453 13.4.1 引言 453 13.4.2 演义 453 13.4.3 基于LINQ的零代码数据访问
13.4.3 层实现 459 13.4.4 LINQ to Provider 462 13.4.5 结论 463 13.5 抢鲜Visual Studio
2008 463 13.5.1 引言 463 13.5.2 Visual Studio 2008概览 464 13.5.3 新特性简介 465 13.5.4
开发示例 465
· · · · · (收起)

你必须知道的.NET 下载链接1

标签

.Net

.NET

编程

C#,

软件开发

ASP.NET

计算机

C

#

评论

大学里读过的,相当相当经典的,可以理论指导实践的书.当时读这本书的时机非常好,里面的问题,正是我百思不得其解的.时机不早不晚.收获极大.适合有一定动手经验的新手读.是一个帮助新手进一步了解.net机制的书.另外cnblogs 也阵中用起来了

许多章节有点繁冗, 总是在说同样的事情。但总体上，还是不错的一本书。

真的还不错,看了两遍的说.

一些基础知识吧

为了生活必须看的书,在.net书籍里头算翘楚了

重读第二遍，书143页的代码有误，父类变量无法调用子类的静态方法，大纲太乱了

略微显得有点虎头蛇尾... 更多是告诉怎么研究.net中的问题 推荐

软件类国内少有的好书，从底层剖析.net

深刻的讲解了dot net框架底层的原理以及设计思想，包括内存管理、设计原则、c#3.0

的新特性以及IL语言等，对底层的接触比较好，推荐读一读。

还行

知其然并知其所以然，本书精髓

大部分内容还是比较实用的，对于开发实战有指导意义，等全部看完在写总结。

知其然之气所以然。易于理解和掌握。

从刚开始接触C#时，就听说过这本书了。奈何那时功力尚浅，匆匆翻了一下，大部分都看不多。如今过去了4年多，自己也一直在注重基础知识的积累。再次翻看，仍然收获不少。

为何只能看一点

原创精品，无需多言，面试必备

非常好的一本书，对于刚入门的同学，推荐买来一读，会对.net有一个框架级别的认识。为以后读clr，提供一定的基础。

[你必须知道的.NET 下载链接1](#)

书评

你必须知道的.net

很长时间没有读书了，回过头来在看看这本书籍，好多当初不理解的东西都理解了，好多设计的原则，面向对象内容，内存管理，各个类型分析等等，是一部难得的上升的书籍，不适合一点net基础也没有的人，适合net的熟悉者想要去提升的人，可以看下底层的东西～

这是一本介绍CLR的书，主要介绍一些底层的东西。买来之后一直在看，跳跃着看的，昨天看到了内存天下这章，还是学到了很多东西，这本书不错，适合有点基础的童鞋看。要想成为高手，肯定要学一些底层的东西，感觉自己还是有很长的路要走，即使看这本书，一下子也不能里边的知识掌...

这本书里面的内容有些杂，从底层IL到OO思想都有涉及，我当时买这本书的理由就是，闲着的时候可以翻翻，挑一些感兴趣的看看。总的来说这是一本可有可无的书，就看你是否对书中的某些专题感兴趣了。

看了两个部分，介绍面向对象的，和IL，内存分配的。介绍面向对象的那部分比较晦涩，感觉作者想尽办法让文章通俗易懂，结果适得其反。介绍IL，内存分配等那部分写得挺好的，很适合不爱读E文的人读。但是，不得不说，想真正了解.NET的底层实现，必须读那几本E文的，比如Essential...

第一次看到如此通俗化的关于.NET介绍的书，极大地加深了对.NET的认识。适合多读几遍会有新的收获。

[你必须知道的.NET 下载链接1](#)