

expert one-on-one™

Wrox Programmer to Programmer™

C# Design and Development

C#设计与开发 专家指南

(美) John Paul Mueller

著

黄敏 王会勇 崔洪斌

译



清华大学出版社

C#设计与开发专家指南

(美) John Paul Mueller 著

黄敏 王会勇 译
崔洪斌

清华大学出版社

北 京

John Paul Mueller
C# Design and Development
EISBN: 978-0-470-41596-2
Copyright © 2009 by Wiley Publishing, Inc.
All Rights Reserved. This translation published under license.

本书中文简体字版由 Wiley Publishing, Inc. 授权清华大学出版社出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

北京市版权局著作权合同登记号 图字: 01-2009-2965

本书封面贴有 Wiley 公司防伪标签, 无标签者不得销售。
版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

C#设计与开发专家指南/(美)缪勒(Mueller, J. P.)著; 黄敏, 王会勇, 崔洪斌 译.

—北京: 清华大学出版社, 2010. 11

书名原文: C# Design and Development

ISBN 978-7-302-23654-2

I. C… II. ①缪…②黄…③王…④崔… III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2010)第 160183 号

责任编辑: 王 军 于 平

装帧设计: 孔祥丰

责任校对: 成凤进

责任印制: 李红英

出版发行: 清华大学出版社

地 址: 北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

邮 编: 100084

社 总 机: 010-62770175

邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者: 清华大学印刷厂

装 订 者: 北京市密云县京文制本装订厂

经 销: 全国新华书店

开 本: 185×260 印 张: 36.5 字 数: 888 千字

版 次: 2010 年 11 月第 1 版 印 次: 2010 年 11 月第 1 次印刷

印 数: 1~4000

定 价: 69.80 元

作者简介

John Mueller 是一名自由作者和技术编辑，他已编写过 82 本书，撰写过 300 余篇文章，主要涉及领域从网络到人工智能，从数据库管理到编程。他最近编写的书涉及 Windows 性能优化、Windows Server 2008 GUI 和 Windows Server 2008 Server 内核(*Windows Server 2008 Server Core*)以及新的 Fluent User Interface (RibbonX)编程人员指南等方面的书籍。他具有娴熟的编辑技术，已帮助超过 58 名作者润色他们所编写的书的内容。John 还对 *DataBased Advisor* 和 *Coast Compute* 两本杂志提供技术编辑支持，并为许多期刊撰写文章，这些期刊包括 *CIO.com*、*DevSource*、*InformIT*、*Informant*、*DevX*、*SQL Server Professional*、*Visual C++ Developer*、*Hard Core VisualBasic*、*asp.netPRO*、*Software Test and Performance* 以及 *Visual Basic Developer*。

John 多才多艺。他在工作之余，经常在工作室度过美好时光，主要工作包括雕刻工艺品或制作蜡烛。他会用整个下午操作车床，或精心地制造书柜，他还喜欢制作甘油香皂。John 也有自己的网站和博客(见 <http://www.johnmuellerbooks.com>)。读者可随时自由访问该网址，并对 John Mueller 的工作提出改进建议。

前 言

任何人经过足够的培训后都能编写代码，但是，只有很少的开发人员能够用代码编写出有价值的应用程序。事实上，许多开发人员很可能甚至不知道有价值的应用程序由什么组成。多年来，开发人员认为有价值的应用程序会包含许多功能，而且用户非常了解应用程序提供的所有灵活性。它就像一盏没有人会忽视的明灯一样，而且它实际上不需要用户的经验，因为经验会影响用户完成有价值的工作。笔者认为：有价值的应用程序就像是灯的开关，任何人都能找到开关，而且大多数人不假思索就能打开灯。事实上，许多人在使用开关时甚至没有认识到他们在使用开关，甚至没有认识到开关的存在。

假设有这样的应用程序，它对用户经验的要求非常简单、非常基本，而且任何人都能够了解这些要求，应用程序能够做用户需要的一切来完成指定的任务，而且还具有快速完成任务所需的速度。读者也许认为这样的应用程序是一个神话，但本书将介绍如何创建这样的应用程序。本书是一本实用的设计书籍。当然，笔者完全违背了某些学术书籍的表现形式，但用户仍对应用程序感到困惑这一事实说明我们需要一本全新的书，这也就是本书的初衷。构造良好的应用程序有助于用户像使用灯的开关那样简单地完成任务，因为此时他们可以不考虑各种经验。

本书介绍的内容能够对作为开发人员的读者提供帮助，特别有助于那些工作于团队环境的开发人员。通过学习本书，可使开发人员在开发时遇到较少的问题，使他们按时完成任务，而且应用程序在工作期间能够满足开发人员的要求。本书介绍了如何完成所有这些任务以及其他任务。例如，会看到有多种设计应用程序的方式，而不只是在学校学到的一些技术。此外，还将介绍用于特殊类型的应用程序的最佳技术，从而使人们更容易地从开始就创建出好的设计。

读者之所以需要学习本书，是因为利用本书能够创建出更好、运行速度更快且问题更少的应用程序，开发人员创建出的应用程序需要很少的维护，而且能够真正实现应用程序所承诺的功能。通过学习本书，可以实现所有这些目标，很难想象在其他书中能够有这样的收获(当然，这里假设读者按照本书给出的建议来获得它能够提供的所有价值)。

本书读者对象

本书适用于编写应用程序的任何人，而且对团队开发也有很大的参考价值。本书介绍的大部分内容也适合咨询师或机构的独立开发人员。编写应用程序的每个人(即便是业余爱好者)均会通过学习本书受益，因为本书适合所有开发人员。虽然书中的某些例子难度较大，但初级和中级开发人员能够掌握本书第 I 部分到第 III 部分介绍的例子(第 IV 部分的复杂例子适合专家级读者)。

本书主要内容

本书主要介绍设计事宜，它涉及读者希望了解的各个方面的内容，如在给定的情形下确定应采用哪一种设计策略。本书还介绍了应用程序的生命周期以及如何保证应用程序在整个生命周期期间得到合理的支持。书中的部分内容专门针对团队开发，如如何选择团队成员以及使团队成员愉快地一起工作。

本书还将介绍一些新的概念。例如，将介绍一个称为性能三角形的新概念。性能三角形由速度、可靠性和安全性组成。在其他书中尚未见到用这一概念来考虑如何创建出确实满足用户需求的具有良好平衡性的应用程序。就用户而言，会看到本书在如何处理用户方面有很多新的概念，其中许多素材来自笔者深入研究、调研的结果以及长期收集的信息。了解实际情况与教师在课堂的理论教学之间的区别很有意义。

本书的第IV部分较为特殊。此部分介绍了以较少的代码创建更好的应用程序时所使用的一些技术。在有些情况下，这些技术能够帮助人们突破.NET Framework的限制。本部分并不适合所有读者，而且人们不会在每个项目中都采用这些技术。这些技术重点针对在特殊项目中要解决的特殊问题。笔者对此部分内容的选择是以过去若干年的调研为基础的。

一个经常要回答的问题是如何找到合适的工具。每个人都知道开发工具的数量要比人们在一生中测试的工具多得多(而且经过一段时间后，所收集的信息就失效了)。本书的第V部分将介绍开发人员在工具方面的需求，并帮助开发人员寻找合适的工具。

本书结构

笔者希望本书易于学习和掌握。许多介绍设计与开发的书籍使得整个设计与开发过程看上去很困难，而且许多书还涉及在真实环境中无法实现的学术方面的问题。读者可以从头到尾学习本书，或将它作为参考。本书按照开发时要考虑的具体问题分成5个部分，且各部分的顺序符合基本的开发过程(从设计到发布这样的过程)顺序。下面简要介绍本书的各个部分。

第I部分：设计策略。此部分重点介绍设计应用程序方面的事宜，但不仅是介绍简单的设计，还介绍了如何构建一支好的开发团队，即如何为自己的应用程序选择恰当的人员。此外，还用较大的篇幅介绍了各相关人员在应用程序中的要求，相关人员是指从构建应用程序的开发人员到管理层、用户以及需要应用程序输出的客户等所有人。最后，介绍了如何创建具有良好平衡性的设计，即设计不会由于不可靠或没有提供足够的安全性而令人失望。

第II部分：编码策略。此部分重点介绍实现过程。已经有了好的设计，那么应该根据设计创建应用程序了。此部分从研究自己的开发环境开始。许多开发人员不会优化他们的工作环境，因此会看到他们的工作过程较为困难。一旦有了好的工作环境，就可以开始编

写代码了。此部分还介绍了调试、测试，以及品质保证过程，最后介绍了部署所完成的应用程序时需要做的工作。在许多书中，部署是最后介绍的内容，但本书还增加了一章，介绍将应用程序投入应用后对应用程序进行维护方面的内容。

第Ⅲ部分：速度、可靠性和安全性。本部分重点介绍构建和维护具有好的平衡性的应用程序时的要求。性能三角形由速度、可靠性和安全性组成，它是应用程序开发和维护过程的基本部分。如果没有保持好的平衡性，则会看到应用程序的性能较差，而且人们不愿意使用它。就维护平衡而言，部分工作属于艺术，部分工作属于科学。本部分将帮助读者了解这两个方面。

第Ⅳ部分：特殊编码方法。笔者从 22 年前大学毕业至今已编写过很多程序，此外，还保存着数百名读者调查以及在线研究大量的素材的记录。本部分介绍开发人员所需要的最常用的编码技术。如前所述，读者也许并不需要这些技术，或者甚至感到不希望使用它们。此部分内容并不适合所有读者，但当需要完成某些似乎不可能完成的任务时，这部分内容确实给出了有价值的建议。

第Ⅴ部分：资源与工具。每个人都需要一些资源和工具来完成他们的编程工作，而且在许多情况下，Visual Studio 环境并没有为开发人员完成其工作提供足够的资源和工具。此部分将回答如何去寻找好的资源或工具方面的问题，将介绍 Microsoft 产品和从第三方得到的产品，还将介绍一些笔者认为有价值的工具。笔者鼓励读者按照本部分介绍的方式去做。用最喜欢的资源和工具来充实自己的工具箱是一件有趣的事。

学习本书需要做的准备工作

为学习本书，读者应在自己的计算机系统中安装 Visual Studio 2008 和 C#。这里假设读者学习本书时使用的是 Windows XP 或 Windows Vista 系统，但书中介绍的内容对于以前版本的 Windows 系统应该也有效(但笔者没有用以前版本的 Windows 测试过本书内容)。

笔者编写本书时使用的是 Visual Studio 2008 Team System。如果读者使用的是不同的版本，则有可能在屏幕上会看到一些不同之处，而且也许会看到菜单的一些微小差别。不要对此担心，你使用的 Visual Studio 2008 版本能够胜任本书的学习。本书的每一名技术编辑均使用了不同版本的产品，并验证了各例子的有效性。此外，笔者还从一些预读者那里收集到了一些版本差异信息，因此，虽然笔者使用的是 Visual Studio 2008 Team System，但各章节都反映了这样的事实，即所介绍的内容也适用于其他版本。

读者还需要知道如何编写 C#代码。本书并不介绍如何编写代码，主要介绍的是如何设计和开发确实有效的应用程序。如果读者没有编写 C#代码的经验，那么学习本书时会有一定的难度。在学习本书前，应该先学习一些介绍编程语言方面的书。

源代码

在读者学习本书中的示例时，可以手工输入所有代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从本书合作站点 <http://www.wrox.com/> 或 www.tupwk.com.cn/downpage 上下载。登录到站点 <http://www.wrox.com/>，使用 Search 工具或使用书名列表就可以找到本书。接着单击本书细目页面上的 Download Code 链接，就可以获得所有的源代码。

注释：

由于许多图书的标题都很类似，所以按 ISBN 搜索是最简单的，本书英文版的 ISBN 是 978-0-470-41596-2。

在下载了代码后，只需用自己喜欢的解压缩软件对它进行解压缩即可。另外，也可以进入 <http://www.wrox.com/dynamic/books/download.aspx> 上的 Wrox 代码下载主页，查看本书和其他 Wrox 图书的所有代码。

勘误表

我们尽全力确保正文或代码中没有错误。然而人无完人，错误总会发生。如果您在我们的书中发现了错误，例如拼写错误或错误的代码段，我们将会非常感激您的反馈。通过递交勘误，您可能会帮助另一名读者避免数小时的受挫，同时，也能帮助我们提供质量更高的书籍。

为了找到本书的勘误页面，请访问 <http://www.wrox.com> 并通过 Search 输入框或根据书名列表找到本书的书名。然后，在本书的详情页面上，单击 Book Errata 链接。在这个页面上您可以看到所有已经为本书提交的并由 Wrox 编辑发布的勘误。

如果在 Book Errata 页上没有找到您所发现的错误，请将错误发送至 wkservice@vip.163.com。我们将会查看信息，如果情况属实，则会发布消息到本书的勘误页，并在本书的后续版本中做出修订。

p2p.wrox.com

如果您希望同作者和本书其他读者进行讨论，可以加入到 <http://p2p.wrox.com> 上的 P2P 论坛。该论坛是一个基于 Web 的系统，您可以在论坛中发布同 Wrox 书籍及相关技术有关的消息，并且可以同其他读者和技术用户交流。论坛提供了订阅特性，可以将论坛上发布的您所感兴趣的话题通过 E-mail 发送给您。论坛中有 Wrox 作者、编辑、其他行业专家以及读者。

在 <http://p2p.wrox.com> 上，您不仅可以找到许多帮助您阅读本书的不同的论坛，而且还可以开发自己的应用程序。要想加入论坛，应执行下列步骤。

- (1) 进入 <http://p2p.wrox.com> 并单击 **Register** 链接。
- (2) 阅读使用条款并单击 **Agree** 按钮。
- (3) 填写加入论坛所要求的信息以及您希望提供的其他可选信息，然后单击 **Submit** 按钮。
- (4) 您将会收到一封电子邮件，其中的内容描述了如何验证您的账号并完成加入过程。

注意：

即使不加入 P2P，您也可以阅读论坛中的消息，但是如果您希望发布自己的消息，则必须加入 P2P。

一旦加入 P2P 之后，您可以发布新的消息并回复其他用户发布的消息。您可以在任何时候阅读 Web 上的消息。如果您希望特定论坛的新的消息以电子邮件的形式发送给您，可单击论坛列表中论坛名字旁边的 **Subscribe to this Forum** 按钮。

要想知道更多关于如何使用 Wrox P2P 的信息，可以阅读 P2P FAQ 中关于论坛软件如何工作以及很多关于 P2P 和 Wrox 书籍的其他常见问题的答案。要想阅读 FAQ，可单击 P2P 页面中的 FAQ 链接。

目 录

第 I 部分 设计策略

第 1 章 定义语言环境	3
1.1 设计战略元素的定义	3
1.2 C#语言	4
1.2.1 C#语言的优点	5
1.2.2 了解劣势	8
1.2.3 使用多种语言开发	10
1.2.4 多平台	11
1.3 盘点工具	11
1.4 收集资源	12
1.4.1 团队语言资源库	12
1.4.2 团队技能	13
1.4.3 经验的意义	13
1.4.4 定义外部资源需求	14
1.5 制定自己的设计策略	14
第 2 章 应用程序生命周期	17
2.1 生命周期的阶段	18
2.1.1 确定应用程序概念	18
2.1.2 构建应用程序开发团队	19
2.1.3 定义规范	21
2.1.4 确定概念的可行性	22
2.1.5 创建设计	23
2.1.6 测试设计	26
2.1.7 实现设计	27
2.1.8 编写应用程序	28
2.1.9 调试和测试应用程序	29
2.1.10 测试用户需求	30
2.1.11 提高可靠性和安全性	31
2.1.12 部署应用程序	32

2.1.13 应用程序的支持与维护	33
2.1.14 应用程序退役	34
2.2 生命周期模型	35
2.2.1 螺旋模型	35
2.2.2 瀑布模型	37
2.2.3 抛弃式模型	39
2.2.4 演化模型	40
2.2.5 增量/迭代模型	41
2.2.6 快速应用开发模型	42
2.3 敏捷编程技术	44
2.3.1 了解敏捷编程	44
2.3.2 敏捷编程的优点	45
2.3.3 敏捷编程的缺点	46
2.3.4 常用敏捷编程策略	46
2.4 制定自己的设计策略	52
第 3 章 定义设计策略	53
3.1 创建对象模型	54
3.1.1 定义类概念	54
3.1.2 功能集	54
3.1.3 定义候选类	55
3.1.4 从候选类中选择类	56
3.1.5 开发内部类	58
3.1.6 添加外部类	58
3.1.7 考虑类的取舍问题	59
3.1.8 实现类设计	60
3.2 创建数据模型	61
3.2.1 定义数据需求	61
3.2.2 访问性、可靠性和安全性需求	63
3.2.3 创建数据类	64
3.3 用户需求	64
3.3.1 定义用户需求	65

3.3.2 创建用户类	67	5.2.1 多线程	101
3.4 将设计转化为 UML	67	5.2.2 速度并不等同于代码	102
3.4.1 Visio	68	5.2.3 多重处理的影响	102
3.4.2 Visio UML 功能概述	69	5.3 测量应用程序的速度	103
3.4.3 由 UML 生成代码	71	5.3.1 标准计数器	103
3.5 制定自己的设计策略	73	5.3.2 定义应用程序计数器	109
第 4 章 设计用户界面	75	5.4 制定自己的设计策略	112
4.1 应用程序窗体类型	75	第 6 章 设计时的可靠性考虑	115
4.1.1 创建对话框实用程序	76	6.1 检验资源的有效性	116
4.1.2 单文档界面	78	6.2 保存数据、设置和状态	118
4.1.3 多文档界面	79	6.2.1 实现数据存储	118
4.1.4 选项卡式界面	80	6.2.2 保存设置的方法	119
4.1.5 开发皮肤界面和自由格式 界面的应用程序	81	6.2.3 实现状态存储	121
4.1.6 RibbonX 应用程序	82	6.3 预测意外情况	121
4.1.7 通知区域	84	6.4 RibbonX 的可靠性优势	122
4.1.8 命令行	85	6.5 制定自己的设计策略	123
4.2 常用用户界面类型	86	第 7 章 设计中的安全性考虑	125
4.2.1 菜单驱动	87	7.1 假设最坏情况	126
4.2.2 任务驱动	87	7.2 消除错误输入	127
4.2.3 信息驱动	88	7.2.1 使用正确的控件或组件	128
4.2.4 向导驱动	88	7.2.2 检验全部用户输入	129
4.2.5 角色驱动	89	7.2.3 验证数据源	135
4.3 制定用户界面策略	90	7.3 隐藏数据	136
4.3.1 使用户保持控制权	91	7.3.1 保持数据本地化	136
4.3.2 使用户沿正确的方向操作	92	7.3.2 加密数据	137
4.3.3 定义交互界面	92	7.3.3 将数据放在受保护的位置	138
4.3.4 定义可靠的界面	93	7.4 添加应用程序监视	138
4.3.5 安全性	93	7.4.1 定义监视策略	139
4.4 可访问性需求	94	7.4.2 实现日志记录	140
4.4.1 视觉元素的考虑	95	7.4.3 使用事件日志	141
4.4.2 添加工具提示和可访问性 说明	96	7.4.4 发送管理员警报	142
4.4.3 色彩使用技巧	97	7.5 团队概述	143
4.5 制定自己的设计策略	98	7.6 制定自己的设计策略	144
第 5 章 设计时的速度考虑	99	第 II 部分 编码策略	
5.1 速度与性能	99	第 8 章 定制集成开发环境	147
5.2 开发高速应用程序	101	8.1 配置 IDE	147

8.1.1 使用 Visual Studio 选项.....	148	10.1.4 Command Window 对话框.....	186
8.1.2 定制工具栏和菜单.....	151	10.2 使用可视化器.....	187
8.1.3 添加外部工具.....	155	10.2.1 为可视化器定义需求.....	187
8.2 使用代码段、宏和插件.....	156	10.2.2 Text 可视化器.....	188
8.2.1 创建和使用代码段.....	157	10.2.3 XML 可视化器.....	189
8.2.2 利用宏自动执行任务.....	159	10.2.4 HTML 可视化器.....	190
8.2.3 依靠插件扩展功能.....	159	10.2.5 DataSet 可视化器.....	191
8.3 使用 Visual Studio 命令行.....	160	10.3 获取第三方可视化器.....	191
8.3.1 访问命令行.....	160	10.4 创建自定义可视化器.....	192
8.3.2 命令行开关.....	161	10.4.1 配置可视化器对象.....	192
8.3.3 执行 Visual Studio 命令.....	164	10.4.2 添加可视化器代码.....	194
8.3.4 运行应用程序.....	165	10.4.3 测试新可视化器.....	196
8.4 为自己的应用程序编写代码.....	165	10.5 为自己的应用程序编写 代码.....	198
第 9 章 脚本.....	167	第 11 章 控件与组件.....	199
9.1 脚本创建选择.....	167	11.1 控件与组件的区别.....	200
9.1.1 使用 SQL Server.....	168	11.2 定义控件类型.....	204
9.1.2 创建 C#脚本.....	168	11.2.1 创建派生控件.....	205
9.1.3 C#表达式评估.....	169	11.2.2 创建 UserControl.....	212
9.1.4 Windows PowerShell.....	169	11.2.3 创建新控件.....	214
9.1.5 语言集成查询.....	170	11.3 创建组件.....	216
9.2 与 SQL Server 交互.....	170	11.3.1 开发新组件时需要考虑 的基本问题.....	216
9.2.1 Visual Studio.....	171	11.3.2 定义新组件项目.....	217
9.2.2 使用 SQL Server 内置功能.....	172	11.4 添加新组件代码.....	218
9.3 在应用程序中加入脚本.....	174	11.5 测试组件.....	222
9.4 C#表达式.....	174	11.6 用对象测试平台测试类.....	224
9.5 制定 Windows PowerShell 解决方案.....	175	11.7 为自己的应用程序编写 代码.....	226
9.5.1 PS 实用程序.....	176	第 12 章 为应用程序编写代码.....	229
9.5.2 创建脚本.....	177	12.1 采用合适的命名习惯.....	229
9.5.3 执行脚本.....	180	12.1.1 匈牙利命名法.....	230
9.5.4 获得调试支持.....	181	12.1.2 驼峰命名法.....	230
9.6 为自己的应用程序编写代码.....	181	12.1.3 帕斯卡命名法.....	231
第 10 章 在 IDE 中浏览数据.....	183	12.1.4 添加专用语法.....	232
10.1 使用 IDE 元素.....	184	12.2 添加命令行功能.....	234
10.1.1 Autos 对话框.....	184	12.2.1 何时使用命令行开关.....	234
10.1.2 Locals 对话框.....	185		
10.1.3 Watch 对话框.....	186		

12.2.2	添加命令行开关	235	14.2.3	错误层次的重要性	279
12.2.3	测试命令行功能	240	14.3	错误处理的一致性	280
12.3	正确退出应用程序	242	14.3.1	单一来源错误	281
12.3.1	在什么情况下只关闭窗体 是不够的	242	14.3.2	多来源错误	281
12.3.2	用 System.Environment 设置退出码	243	14.3.3	最终错误处理	281
12.3.3	Application.Exit()方法	245	14.4	设计自恢复应用程序	282
12.3.4	用 P/Invoke 退出 应用程序	247	14.5	为自己的应用程序编写 代码	284
12.4	使用自定义功能	248	第 15 章	测试、调试和质量保证	285
12.4.1	何时使用自定义功能	248	15.1	测试应用程序	285
12.4.2	使用自定义特征和 异常类	249	15.1.1	创建新测试	286
12.5	为自己的应用程序 编写代码	249	15.1.2	运行测试	290
第 13 章	文档	251	15.1.3	利用 Test View 对话框 设置测试属性	291
13.1	对应用程序增加文档支持	251	15.1.4	利用测试列表编辑器 组织测试	293
13.1.1	文档概述	252	15.1.5	Code Coverage Results 对话框	294
13.1.2	考虑开发人员的需求	253	15.1.6	Test Runs 对话框	296
13.1.3	考虑用户需求	253	15.1.7	人工测试	297
13.1.4	考虑管理员的需要	255	15.2	调试应用程序	298
13.2	创建文档注释	256	15.2.1	调试对话框	298
13.2.1	注释的类型	256	15.2.2	向 IDE 提问题	299
13.2.2	注释的正确位置	259	15.2.3	浏览内存内容	300
13.3	使用 XML 文件	261	15.2.4	通过用户获得漏洞	301
13.4	文档输出的其他作用	265	15.3	质量保证检验	301
13.4.1	用 LINQ 查询文档文件	265	15.3.1	与 B 团队交流	301
13.4.2	根据文档自动创建 应用程序报告	266	15.3.2	构造有价值的反馈	302
13.5	为自己的应用程序 编写代码	266	15.3.3	用户参与	303
第 14 章	错误处理	267	15.4	为自己的应用程序编写 代码	303
14.1	对错误处理不做假设的 优点	268	第 16 章	部署应用程序	305
14.2	特殊错误处理的重要性	269	16.1	确定部署类型	306
14.2.1	正确地使用标准异常类	269	16.1.1	测试版	306
14.2.2	实现自定义错误处理	278	16.1.2	局部部署	307
			16.1.3	企业部署	308
			16.1.4	热塑-包装部署	309
			16.1.5	主版本和次版本	309

16.2	选择部署方法	311	18.2.2	可靠性	343
16.2.1	利用 XCopy 部署	312	18.2.3	安全性	344
16.2.2	创建基本的安装项目	313	18.3	为应用程序创建完美的 性能三角形	346
16.2.3	修改细节	319	18.4	使应用程序既可靠又安全	347
16.2.4	使用项目安装向导	320	第 19 章	应用程序的速度	349
16.2.5	CAB 方法	321	19.1	定义速度的平衡性	349
16.3	选择介质类型	322	19.1.1	在错误的位置删除代码	350
16.4	完成部署	323	19.1.2	创建错误的答案确实快	351
16.5	评价结果	325	19.2	速度测量	352
16.6	为自己的应用程序编写 代码	326	19.2.1	示例代码简介	353
第 17 章	应用程序的支持与维护	327	19.2.2	启动性能向导	354
17.1	创建错误日志	327	19.2.3	测评性能	357
17.1.1	定义错误	327	19.2.4	浏览性能结果	363
17.1.2	处理错误日志设置	328	19.2.5	性能报告比较	366
17.1.3	创建错误日志类	329	19.3	使应用程序既可靠又安全	368
17.1.4	浏览错误日志输出	332	第 20 章	应用程序的可靠性	369
17.2	利用错误日志提供支持	333	20.1	定义可靠性的平衡	369
17.2.1	定义错误、资源和 解决方案	333	20.1.1	避免应用程序过度保守	370
17.2.2	确定是否需要开发人员 直接交互	334	20.1.2	创建绝对不做任何无效 工作的应用程序	370
17.2.3	创建支持日志	334	20.1.3	定义外部威胁	372
17.3	通过错误日志和支持日志 实现维护	334	20.1.4	制定可靠性计划	372
17.3.1	维护优先级	335	20.2	意想不到的用户	373
17.3.2	检验维护要求	335	20.2.1	最好是在测试期间中断 应用程序	374
17.3.3	测试维护	335	20.2.2	创建经验教训日志	374
17.3.4	部署补丁	336	20.3	检验代码	375
17.4	为自己的应用程序编写 代码	336	20.3.1	运行代码分析	376
			20.3.2	计算代码度量	377
			20.4	使应用程序既可靠又安全	379
第 III 部分	速度、可靠性及安全性		第 21 章	应用程序的安全性	381
第 18 章	围绕性能三角形开发	339	21.1	定义安全性的平衡	381
18.1	初始开发阶段后的工作	340	21.1.1	使数据远离用户	382
18.2	定义性能三角形	340	21.1.2	创建性能障碍	382
18.2.1	速度	342	21.1.3	过度监管	383
			21.2	入侵应用程序	384

21.2.1	依靠内部测试员测试	385
21.2.2	第三方测试	386
21.2.3	检验不可能性	387
21.2.4	交互作用	388
21.3	更新安全性配置文件	388
21.4	管理监视过程	389
21.5	数据的加密与解密	390
21.6	使应用程序既可靠又安全	391

第IV部分 编码技术

第22章 序列化 XML 395

22.1	以 XML 作为存储方法	396
22.1.1	实现 XCopy 应用程序	396
22.1.2	XML 数据示例	397
22.1.3	在报告中使用 XML 数据	399
22.2	创建 XML 存储类	400
22.2.1	定义 XML 数据元素	400
22.2.2	创建自定义数据说明	401
22.2.3	设计健壮的属性	402
22.3	以 XML 格式保存数据	403
22.3.1	选择数据位置	403
22.3.2	实现保存例程	404
22.4	以 XML 格式加载数据	405
22.4.1	从磁盘获得 XML 数据	406
22.4.2	使用 XML 数据修改环境	408
22.5	处理序列化的 XML 数据	409
22.5.1	添加新数据	409
22.5.2	编辑已有项	411
22.5.3	删除数据项	413
22.5.4	对数据项排序	415
22.5.5	在屏幕上显示数据项	417
22.6	实现自己的设计	417

第23章 LINQ 419

23.1	Visual Studio 2008 对 LINQ 的支持	419
------	-------------------------------	-----

23.1.1	LINQ 及提供程序	420
23.1.2	LINQ 与 SQL 的比较	420
23.1.3	使用 LINQ 实现非查询解决方案	424
23.2	定义基本操作符	424
23.2.1	操作符 from 和 select	424
23.2.2	操作符 where	425
23.2.3	操作符 orderby	426
23.3	创建 LINQ 查询	427
23.3.1	定义简单查询	427
23.3.2	多数据源的连接	428
23.3.3	操作符 let	430
23.4	用 LINQ 与各种数据源交互	431
23.4.1	内置提供程序	431
23.4.2	LINQ to Objects 提供程序	432
23.4.3	LINQ to DataSet 提供程序	435
23.4.4	LINQ to SQL 提供程序	439
23.4.5	LINQ to XML 提供程序	444
23.4.6	第三方提供程序	449
23.5	实现自己的设计	451

第24章 利用 F#使应用程序增值 453

24.1	了解 F#	454
24.1.1	函数式语言	454
24.1.2	定义函数式语言的需求	455
24.2	Visual Studio 2008 对 F#的支持	455
24.2.1	下载产品	456
24.2.2	安装 F#	456
24.3	使用命令行	457
24.3.1	修改命令行路径来满足 F#的要求	457
24.3.2	定义基本脚本	458
24.3.3	运行应用程序	458
24.3.4	与 F#进行交互	459
24.4	创建 F#应用程序	460

24.5	实现自己的设计	462
第 25 章	创建多线程应用程序	463
25.1	了解多线程	463
25.1.1	定义多线程	464
25.1.2	多线程的优点	465
25.1.3	多线程的缺点	467
25.1.4	多线程的现状	468
25.2	创建基本的多线程应用 程序	469
25.2.1	定义项目	469
25.2.2	添加文件处理线程类	470
25.2.3	添加监视线程类	471
25.2.4	开发应用程序代码	475
25.2.5	测试多线程应用程序	478
25.3	实现自己的设计	481
第 26 章	创建报告和其他输出	483
26.1	定义优秀报告的元素	484
26.1.1	以可接受的报告作为 工作的起点	484
26.1.2	将可接受的报告提升 到好的水平	485
26.1.3	测试报告	486
26.2	了解用户报告需求	486
26.2.1	内容需求	487
26.2.2	定义格式需求	487
26.2.3	包含机构策略	488
26.2.4	为报告建立用户关系	489
26.3	创建自修改报告	490
26.4	设计其他形式的应用 程序输出	491
26.5	实现自己的设计	493
第 27 章	设计底层应用程序元素	495
27.1	定义 P/Invoke	495
27.2	调用外部函数	497
27.2.1	添加所需目录	498
27.2.2	提供结构、枚举和常数	498
27.2.3	创建外部函数引用	501

27.2.4	使用函数调用数据	502
27.2.5	在托管代码中使用外部 函数	503
27.2.6	提供错误处理	505
27.3	设计数据结构	510
27.3.1	为标准控制台句柄 创建枚举	510
27.3.2	创建基本数据结构	511
27.3.3	定义函数调用	512
27.3.4	在托管代码中采用 数据结构	514
27.4	实现自己的设计	515

第V部分 资源与工具

第 28 章	选择资源与工具	519
28.1	结束耗时的混乱	519
28.1.1	创建档案	521
28.1.2	有效地对资源和工具 升级	522
28.1.3	与供应商建立良好 关系	523
28.2	定义有价值的资源	524
28.3	定义有价值的工具	524
28.4	创建团队工具箱	525
28.4.1	了解团队的要求	526
28.4.2	定义使用要求和策略	526
28.5	获取资源与工具	527
第 29 章	Microsoft 资源与工具	529
29.1	寻找 Microsoft 资源与工具	529
29.1.1	MSDN	530
29.1.2	TechNet	530
29.1.3	使用 Microsoft Download	531
29.1.4	微软搜索	532
29.1.5	在线查找资源和工具	533
29.2	用 XML Notepad 2007 编辑 XML 文件	535

29.2.1	获得 XML Notepad 2007	535
29.2.2	第一次使用 XML Notepad 2007	535
29.2.3	修改文件	536
29.2.4	使用 XSLT 文件	536
29.2.5	比较 XML 文件	537
29.3	利用 Microsoft PowerCommands 提高 Visual Studio 2008 的 性能	537
29.4	获取资源与工具	542
第 30 章	第三方资源与工具	545
30.1	寻找第三方资源与工具	545
30.2	用 Vischeck 测试应用程序 颜色	549
30.3	利用 FxCop 提高代码的 可靠性和一致性	552
30.3.1	获得 FxCop	552
30.3.2	使用 FxCop GUI	553
30.3.3	修复错误	555
30.3.4	考虑规则	558
30.3.5	在构建期间使用 FxCop ..	558
30.4	利用 LINQPad 创建 LINQ 查询	560
30.5	获取资源与工具	562