

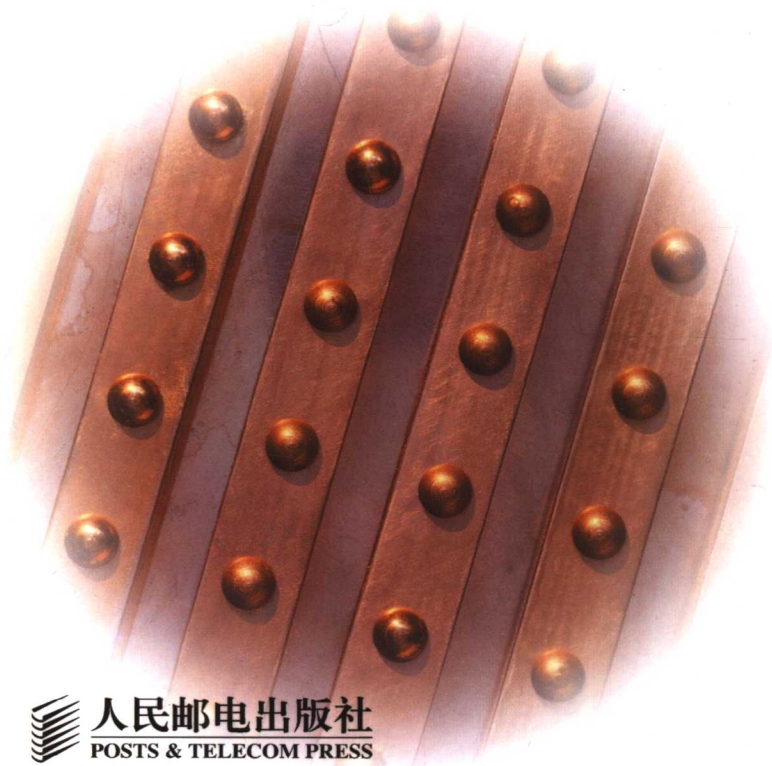
 WILEY

Visual Basic 设计模式

Visual Basic Design Patterns

[美] Mark Grand Brad Merrill 著

杨环英 周锐博 译



 人民邮电出版社
POSTS & TELECOM PRESS



Visual Basic

设计模式

[美] Mark Grand Brad Merrill 著

杨环英 周锐博 译

人民邮电出版社

图书在版编目 (CIP) 数据

Visual Basic 设计模式 / (美) 格朗德 (Grand, M.) 等著; 杨环英, 周锐博译.
—北京: 人民邮电出版社, 2006.8

ISBN 7-115-15015-X

I. V... II. ①格... ②杨... ③周... III. BASIC 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2006) 第 078283 号

版 权 声 明

Visual Basic Design Patterns

Copyright © 2005 by John Wiley & Sons, Ltd.

All right reserved.

Authorized translation form the English language edition published by John Wiley & Sons, Inc.

本书中文简体字版由 John Wiley & Sons 公司授权人民邮电出版社出版, 专有出版权属于人民邮电出版社。

Visual Basic 设计模式

-
- ◆ 著 [美] Mark Grand Brad Merrill
译 杨环英 周锐博
责任编辑 陈 昇
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销
 - ◆ 开本: 800×1000 1/16
印张: 30.5
字数: 684 千字 2006 年 8 月第 1 版
印数: 1—5 000 册 2006 年 8 月北京第 1 次印刷

著作权合同登记号 图字: 01-2003-2749 号

ISBN 7-115-15015-X/TP · 5565

定价: 59.00 元

读者服务热线: (010) 67132705 印装质量热线: (010) 67129223

内容提要

本书介绍了 Visual Basic 软件设计中的常见模式。全书共分为 8 章, 内容涉及 UML 概述、软件生命周期、基础设计模式、创建模式、划分模式、结构型模式、行为模式和并发模式。对于设计模式初学者和具有丰富经验的程序员, 本书均具有一定的参考价值。

作者简介

Mark Grand 是亚特兰大地区的一名分布式系统面向对象设计和 Java 顾问。他是第一代互联网商业 B2B 电子商务产品的架构师。在他从事 Java 工作之前，他从事了 11 年多的 4GL 设计师和实现师。他的最新职业为电子数据交换产品的架构师和项目经理。**Mark** 就职于很多 MIS 组织，职位包括软件架构师、数据库架构师和网络设计师等。

自 1982 年后，**Mark** 开始涉及面向对象编程和设计。现在，他因为编著模式方面的畅销书而闻名。**Mark** 曾在加利福尼亚大学伯克利分校、Sun 公司和其他组织讲学。从 <http://www.markgrand.com/> 可以找到 **Mark Grand** 的更多信息。

Brad Merrill 现在是微软 ASP.NET 团队的软件工程师。他以前是微软的 .NET 技术宣传师、Sybase 和 Digital Equipment 公司的软件工程师。他的专业领域为分布式系统、事务处理、操作系统和编译器技术。**Brad** 生活在华盛顿的雷蒙德。他是一名热情的国际象棋选手和桥牌选手。他的联系方式为 zbrad@cybercom.net，或从 <http://www.cybercom.net/~zbrad> 了解相关信息。

首先，读者要理解什么是设计模式及其重要性。设计模式是具有高度重用性、经过分门别类、得到广泛认可的一套软件设计经验。可见，掌握设计模式后，不仅能缩短软件开发周期，还能提高软件的质量和可重用性，程序员之间通过理论化的模式描述，也更加容易进行技术交流。

其次，读者要明白，设计模式是无数程序员在实践中的经验总结和观察所得。不同领域的程序员可能从不同的角度发现新的设计模式。同时，模式的数量必定会不断增加。因此，读者最重要的是要掌握基本设计模式，才能对其他设计模式触类旁通。对于初学者，尤其如此。

本书主要介绍 Visual Basic .NET 中的常见设计模式。对每一种模式，均从模式名称、要求、解决方案、实现、结论、.NET 用法、代码示例、相关模式几个方面进行介绍。

本书的作者具有丰富的软件开发经验，而且是某些模式设计畅销书的作者。

本书具有以下特点：

- 本书介绍的设计模式均是 Visual Basic .NET 中的常见设计模式；
- 对于所有设计模式，作者均用 UML 进行了准确的表达，便于读者尽快掌握所介绍的模式；
- 对每种模式，本书提供了完整的案例。便于读者尽快应用于实践之中。

无论对于设计模式初学者，还是具有丰富软件设计经验的程序员，都可从本书中获得丰富的模式设计知识。

由于该书内容涉及面广，且译者水平有限、时间仓促，错误之处在所难免，希望广大读者不吝指正。联系方式 E-mail: web_zhou@21cn.com。

译者

2006 年 5 月

丰富的经验赋予程序员无穷的智慧。因为程序员积累经验后，可以识别与先前已解决的问题相类似的新问题。拥有更丰富经验后，可以发现相似问题的解决方案可遵循相同的模式。掌握这些模式后，有经验的程序员可以判断出这些模式适用的场合，并可以立即应用相应的解决方案，而不必停下来分析问题、制定可行策略。

软件模式是针对软件开发过程中，一些重复问题的可重用解决方案。程序员发现一种模式时，这体现了程序员的洞察力。在大部分情况下，从难以用语言表达的洞察到成熟的思想，程序员极难用语言进行清晰表达。但这也是极其有价值的一步。当我们对模式拥有足够的了解，可以进行明确表达时，就可以将该模式与其他模式巧妙结合。更重要的是，将模式理论化后，了解该模式的程序员可以讨论和使用该模式。这允许程序员更有效地协作，允许运用他们的聪明才智。还有助于避免程序员对某个问题的不同解决方案争论不休，但实际上，程序员考虑的是同一个解决方案，只不过表达方式不同而已。

对于经验不够丰富、尚未了解模式的程序员来说，将模式理论化还有另外一个好处。将模式理论化后，经验丰富的程序员可以向不知道该模式的程序员进行讲解。

本书的价值在于，为有经验的程序员提供了通用的理论化表达方式。本书还让不了解模式的程序员对模式有更多的了解。

本书介绍了大量模式，还有其他一些作者已知的模式，但由于时间有限，本书并未涉及。读者也可以自己探寻其他模式。读者发现的模式可能专业性很强，只有少数人感兴趣。有些模式可能对很多人有用，因此，值得收集在本书的下一卷中。如果读者希望与本书作者探讨这样的模式，请发送电子邮件至 mgrand@mindspring.com。

本书列出的模式包含一些建设性的方式，可用于组织软件开发周期中的各部分。还有其他一些模式，采用它们可能导致一些组织性不强的程序。这些模式称作反模式，因为反模式可能消除了模式的优势。本书不打算介绍这些反模式，相关介绍可阅读其他参考书。对反模式感兴趣的读者可以阅读参考文献[BMMM98]。

注意 因为本书内容只与软件模式相关，所以在本书的其余部分将简称其为模式。

模式简史

软件模式的思想来源于建筑学领域。一个名叫 Christopher Alexander 的建筑师曾经编撰了一本书,描述了建造建筑物和城市规划中的一些模式。类似的书籍有: *A Pattern Language: Towns, Buildings, Construction* (牛津大学出版社出版) 和 *The Timeless Way of Building* (牛津大学出版社出版)。

本书提出的思想可适用于建筑学之外的许多领域, 包括软件。

1987 年, Ward Cunningham 和 Kent Beck 使用 Alexander 的某些思想提出了 5 种模式, 用于指导用户界面的设计。他们在 OOPSLA-87 上发表了一篇相关论文。论文题目是“在面向对象程序中使用模式语言”。

在 20 世纪 90 年代早期, Erich Gamma、Richard Helm、John Vlissides 和 Ralph Johnson 这 4 位作者开始合作撰写一本影响广泛的书籍《*Design Patterns (设计模式)*》(Addison-Wesley 出版社出版)。它推广了模式的思想, 它对本书具有最大的影响。《*Design Patterns (设计模式)*》一书通常称作“四雄之书”或 GoF。该书的示例使用 C++ 和 Smalltalk 语言, 因为编写该书时, 还不存在 UML。现在, UML 被广泛接受, 作为面向对象设计的首选记号法, 因此, 本书采用了 UML。本书的示例中使用 Visual Basic .NET 语言。本书对模式的描述均是从 VB.NET 的角度来阐述。

本书的组织

本书着重介绍在宏观架构级别使用的设计模式。前两章提供了帮助读者理解后面章节所介绍模式的相关材料。

- 第 1 章首先描述本书使用的 UML 子集。
- 第 2 章概述软件生命周期, 提供模式使用的上下文。本章还提供了一个案例, 用于演示可应用设计模式的场合。

其余章节描述各种不同的模式。

- 第 3 章介绍一些基本模式, 它们在本质上具有基础性。这些模式适用于各种不同场合。还可应用于其他模式之中。
- 第 4 章介绍如何组织不同场合所创建对象的模式。
- 第 5 章介绍使用分而治之途径解决问题的模式。
- 第 6 章介绍的模式描述了将对象组合成不同结构类型的方式。
- 第 7 章介绍的模式描述了组织行为的不同方式。
- 第 8 章介绍管理并发的模式。

模式描述

本书介绍的模式使用特定的格式来描述。该格式包括如下信息。

- 模式的常用名。若该模式还具有更多名称，则同时给出它的其他名称。
- 对问题的描述，包括一个具体示例以及该具体问题的解决方案。
- 考虑因素或要求的小结。这些因素导致通用解决方案的形成，或者避免采用不适宜的解决方案。
- 通用解决方案。
- 使用已知解决方案解决问题的结论（包括正反两方面）。
- 列出相关模式。

其他模式书籍在介绍上述内容时，可能方式有所不同。本书的模式均与设计阶段相关。在本书中，与设计阶段相关模式的介绍按以下各小节进行组织。

模式名称

这一节的标题包含模式的名称。在标题下，可能还提供一个参考书目，它指出了模式的来源。在该标题下，大部分模式没有更多的正文。因此，这一节有时包含相应模式的其他名称，或包含相应模式的派生性质与一般性质的相关信息。

概要

这一节包含模式的简要描述。概要中介绍了模式提供的解决方案的实质。概要主要供有经验的程序员参考，他们可以将讲述的模式与已经知道、但不知名称的模式相对应。

如果根据名称和概要不能识别模式，请不要灰心。应当仔细阅读这一模式描述的剩下内容，努力理解该模式。

背景

这一节描述模式所解决的问题。对于大部分模式，以具体的示例提出问题。提出具体问题之后，再为具体问题提出设计解决方案。

要求

这一节小结应用模式的所有考虑因素，这些因素将形成下一节提出的通用解决方案。也可能会介绍不使用模式的原因。使用或不使用解决方案的原因以项目符号的形式给出。

- ⊙ 使用解决方案的原因以笑脸项目符号的形式给出。
- ⊗ 不使用解决方案的原因以悲伤表情项目符号的形式给出。

解决方案

这一节是模式的核心。它介绍了模式所解决问题的通用解决方案。

实现

这一节讲解实现解决方案时，应当考虑的重要因素。还可能介绍简化解决方案的一些常

见修改办法。

结论

这一节解决使用解决方案的含意、优缺点。大部分结论按如下所示的项目符号方式进行组织。

- ☺ 正面结论以笑脸项目符号的形式给出。
- 中性结论以圆点项目符号的形式给出（该项目符号还用于标准项目符号列表中）。
- ⊗ 反面结论以悲伤表情项目符号的形式给出。

.NET 用法

若在.NET 框架中存在适当的模式示例，这一节将作相应介绍。在.NET 框架中未使用该模式时，则不包含这一节内容。

代码示例

这一节包含代码示例，它演示使用模式进行设计的实现示例。通常，这一节实现的设计是前面“背景”一节中描述的设计。

相关模式

这一节包含与所讨论模式相关的模式列表。

本书的读者对象

对于使用.NET 环境的各层次程序员来说，本书均具有参考价值。具有不同经验的程序员，从本书中可以各取所需。经验较少的程序员可着重学习每种模式解决的问题及其解决方案。

经验较丰富的程序员，他们已经熟悉了模式中描述的问题及其解决方案，这些人可从使用或不使用解决方案的讨论以及结论中汲取知识营养。

各层次程序员都将从解决方案的理论化描述中获益。那些已经熟悉 UML 或软件生存周期的人，可以跳过这些章节。

可按顺序阅读讲述模式的各章。有些人选择性地阅读某些模式，这些模式可能与他们承担的任务有直接联系。团队中的程序员在阅读本书时，也可以组织一个讨论组，相互分享从学习这些模式所获得的成果，每个人都可从中受益。在很多地方，也存在这样的模式讨论组。讨论组列表可参见 <http://c2.com/cgi/wiki?PatternsGroups>。

关于本书网站

从 <http://www.wiley.com/go/vbdesignpatterns> 可访问本书的网站。它包含本书所介绍模式的概述，还包含本书列出的所有代码示例。

目 录

第 1 章 UML 概述	1	3.2.1 概要	44
1.1 类图	1	3.2.2 背景	44
1.2 协作图	12	3.2.3 要求	45
1.3 状态图	19	3.2.4 解决方案	45
第 2 章 软件生命周期	21	3.2.5 实现	46
案例研究	23	3.2.6 结论	47
2.1.1 商业案例	23	3.2.7 .NET 用法	47
2.1.2 定义需求规范书	24	3.2.8 代码示例	47
2.1.3 开发高层基本用例	25	3.2.9 相关模式	48
2.1.4 面向对象分析	26	3.3 抽象基类	49
2.1.5 面向对象设计	28	3.3.1 概要	49
第 3 章 基础设计模式	37	3.3.2 背景	49
3.1 委托（未使用继承时）	37	3.3.3 要求	50
3.1.1 概要	37	3.3.4 解决方案	50
3.1.2 背景	37	3.3.5 实现	51
3.1.3 要求	40	3.3.6 结论	52
3.1.4 解决方案	41	3.3.7 .NET API 用法	52
3.1.5 实现	41	3.3.8 代码示例	52
3.1.6 结论	41	3.3.9 相关模式	55
3.1.7 .NET 用法	42	3.4 接口和抽象类	55
3.1.8 代码示例	42	3.4.1 概要	55
3.1.9 相关模式	44	3.4.2 背景	55
3.2 接口	44	3.4.3 要求	55
		3.4.4 解决方案	55
		3.4.5 结论	56

3.4.6	.NET API 用法	56	4.2.2	背景	88
3.4.7	代码示例	57	4.2.3	要求	89
3.4.8	相关模式	58	4.2.4	解决方案	89
3.5	不变模式	59	4.2.5	实现	91
3.5.1	概要	59	4.2.6	结论	91
3.5.2	背景	60	4.2.7	代码示例	92
3.5.3	要求	60	4.2.8	相关模式	97
3.5.4	解决方案	61	4.3	构建器	97
3.5.5	实现	62	4.3.1	概要	97
3.5.6	结论	62	4.3.2	背景	97
3.5.7	.NET API 用法	62	4.3.3	要求	100
3.5.8	代码示例	62	4.3.4	解决方案	100
3.5.9	相关模式	63	4.3.5	实现	102
3.6	代理	63	4.3.6	结论	103
3.6.1	概要	64	4.3.7	.NET API 用法	103
3.6.2	背景	64	4.3.8	代码示例	104
3.6.3	要求	65	4.3.9	相关模式	105
3.6.4	解决方案	65	4.4	原型	105
3.6.5	实现	65	4.4.1	概要	105
3.6.6	结论	66	4.4.2	背景	105
3.6.7	代码示例	66	4.4.3	要求	106
3.6.8	相关模式	73	4.4.4	解决方案	107
第 4 章	创建模式	75	4.4.5	实现	108
4.1	工厂方法	76	4.4.6	结论	108
4.1.1	概要	76	4.4.7	.NET API 用法	109
4.1.2	背景	76	4.4.8	代码示例	109
4.1.3	要求	78	4.4.9	相关模式	113
4.1.4	解决方案	78	4.5	单件	114
4.1.5	实现	79	4.5.1	概要	114
4.1.6	结论	81	4.5.2	背景	114
4.1.7	.NET API 用法	81	4.5.3	要求	115
4.1.8	代码示例	82	4.5.4	解决方案	115
4.1.9	相关模式	87	4.5.5	实现	116
4.2	抽象工厂	87	4.5.6	结论	118
4.2.1	概要	87	4.5.7	.NET API 用法	118
			4.5.8	代码示例	118

4.5.9 相关模式	119	5.3.4 解决方案	154
4.6 对象池	120	5.3.5 实现	155
4.6.1 概要	120	5.3.6 结论	155
4.6.2 背景	120	5.3.7 代码示例	155
4.6.3 要求	122	5.3.8 相关模式	157
4.6.4 解决方案	122	第 6 章 结构型模式	159
4.6.5 实现	124	6.1 适配器	159
4.6.6 结论	126	6.1.1 概要	159
4.6.7 代码示例	126	6.1.2 背景	159
4.6.8 相关模式	131	6.1.3 要求	161
第 5 章 划分模式	133	6.1.4 实现	162
5.1 过滤器	133	6.1.5 结论	163
5.1.1 概要	133	6.1.6 代码示例	163
5.1.2 背景	133	6.1.7 相关模式	169
5.1.3 要求	134	6.2 迭代器	169
5.1.4 解决方案	134	6.2.1 概要	169
5.1.5 实现	137	6.2.2 背景	169
5.1.6 结论	137	6.2.3 要求	170
5.1.7 .NET API 用法	137	6.2.4 解决方案	170
5.1.8 代码示例	138	6.2.5 实现	171
5.1.9 相关模式	142	6.2.6 结论	172
5.2 复合	142	6.2.7 .NET API 用法	173
5.2.1 概要	142	6.2.8 代码示例	173
5.2.2 背景	142	6.2.9 相关模式	175
5.2.3 要求	144	6.3 桥接	175
5.2.4 解决方案	144	6.3.1 概要	175
5.2.5 实现	145	6.3.2 背景	175
5.2.6 结论	146	6.3.3 要求	177
5.2.7 .NET 用法	147	6.3.4 解决方案	178
5.2.8 代码示例	147	6.3.5 实现	179
5.2.9 相关模式	152	6.3.6 结论	179
5.3 只读接口	152	6.3.7 .NET API 用法	180
5.3.1 概要	152	6.3.8 示例	180
5.3.2 背景	152	6.3.9 相关模式	185
5.3.3 要求	154	6.4 外观	185

6.4.1	概要	185	6.7.7	代码示例	217
6.4.2	背景	186	6.7.8	相关模式	219
6.4.3	要求	187	6.8	修饰器	219
6.4.4	解决方案	187	6.8.1	概要	220
6.4.5	实现	188	6.8.2	背景	220
6.4.6	结论	188	6.8.3	要求	222
6.4.7	.NET API 用法	188	6.8.4	解决方案	222
6.4.8	示例	189	6.8.5	实现	223
6.4.9	相关模式	192	6.8.6	结论	223
6.5	享元	192	6.8.7	代码示例	223
6.5.1	概要	192	6.8.8	相关模式	226
6.5.2	背景	192	6.9	缓存管理	226
6.5.3	要求	196	6.9.1	概要	226
6.5.4	解决方案	196	6.9.2	背景	226
6.5.5	实现	197	6.9.3	要求	227
6.5.6	结论	197	6.9.4	解决方案	228
6.5.7	.NET API 用法	198	6.9.5	实现	229
6.5.8	示例	198	6.9.6	结论	233
6.5.9	相关模式	203	6.9.7	代码示例	234
6.6	动态链接	203	6.9.8	相关模式	245
6.6.1	概要	203	第 7 章	行为模式	247
6.6.2	背景	203	7.1	责任链	247
6.6.3	要求	205	7.1.1	概要	247
6.6.4	解决方案	205	7.1.2	背景	248
6.6.5	实现	206	7.1.3	要求	248
6.6.6	结论	207	7.1.4	解决方案	249
6.6.7	.NET API 用法	208	7.1.5	实现	250
6.6.8	代码示例	208	7.1.6	结论	251
6.6.9	相关模式	212	7.1.7	.NET API 用法	251
6.7	虚拟代理	212	7.1.8	代码示例	251
6.7.1	概要	212	7.1.9	相关模式	256
6.7.2	背景	212	7.2	命令	256
6.7.3	要求	214	7.2.1	概要	257
6.7.4	解决方案	214	7.2.2	背景	257
6.7.5	实现	216	7.2.3	要求	258
6.7.6	结论	217			

7.2.4	解决方案	258	7.6.2	背景	319
7.2.5	实现	259	7.6.3	要求	320
7.2.6	结论	261	7.6.4	解决方案	320
7.2.7	.NET API 用法	261	7.6.5	实现	321
7.2.8	代码示例	261	7.6.6	结论	323
7.2.9	相关模式	267	7.6.7	.NET API 用法	324
7.3	小语言	267	7.6.8	代码示例	324
7.3.1	概要	268	7.6.9	相关模式	326
7.3.2	背景	268	7.7	状态	327
7.3.3	要求	277	7.7.1	概要	327
7.3.4	解决方案	277	7.7.2	背景	327
7.3.5	实现	278	7.7.3	要求	330
7.3.6	结论	279	7.7.4	解决方案	330
7.3.7	.NET API 用法	280	7.7.5	实现	332
7.3.8	代码示例	280	7.7.6	结论	332
7.3.9	相关模式	291	7.7.7	代码示例	333
7.4	中介者	292	7.7.8	相关模式	338
7.4.1	概要	292	7.8	策略	338
7.4.2	背景	292	7.8.1	概要	339
7.4.3	要求	294	7.8.2	背景	339
7.4.4	解决方案	294	7.8.3	要求	339
7.4.5	实现	295	7.8.4	解决方案	340
7.4.6	结论	297	7.8.5	实现	340
7.4.7	代码示例	297	7.8.6	结论	341
7.4.8	相关模式	302	7.8.7	.NET API 用法	341
7.5	快照	302	7.8.8	代码示例	341
7.5.1	概要	302	7.8.9	相关模式	343
7.5.2	背景	303	7.9	空对象	343
7.5.3	要求	306	7.9.1	概要	344
7.5.4	解决方案	306	7.9.2	背景	344
7.5.5	实现	309	7.9.3	要求	345
7.5.6	结论	316	7.9.4	解决方案	345
7.5.7	代码示例	316	7.9.5	实现	346
7.5.8	相关模式	318	7.9.6	结论	346
7.6	观察者	318	7.9.7	代码示例	346
7.6.1	概要	318	7.9.8	相关模式	348

7.10 模板方法	348	8.1.7 代码示例	379
7.10.1 概要	348	8.1.8 相关模式	381
7.10.2 背景	348	8.2 静态锁定顺序 (Static Locking Order)	382
7.10.3 要求	350	8.2.1 概要	382
7.10.4 解决方案	350	8.2.2 背景	382
7.10.5 实现	351	8.2.3 要求	382
7.10.6 结论	351	8.2.4 解决方案	383
7.10.7 代码示例	351	8.2.5 结论	383
7.10.8 相关模式	354	8.2.6 实现	384
7.11 访问者	354	8.2.7 已知的应用	384
7.11.1 概要	354	8.2.8 代码示例	384
7.11.2 背景	354	8.2.9 相关模式	386
7.11.3 要求	357	8.3 锁对象 (Lock Object)	387
7.11.4 解决方案	357	8.3.1 概要	387
7.11.5 实现	359	8.3.2 背景	387
7.11.6 结论	360	8.3.3 要求	387
7.11.7 代码示例	361	8.3.4 解决方案	388
7.11.8 相关模式	364	8.3.5 实现	389
7.12 哈希适配器对象	364	8.3.6 结论	390
7.12.1 概要	364	8.3.7 代码示例	390
7.12.2 背景	364	8.3.8 相关模式	392
7.12.3 要求	367	8.4 受保护的挂起	392
7.12.4 解决方案	368	8.4.1 概要	392
7.12.5 实现	369	8.4.2 背景	392
7.12.6 结论	370	8.4.3 要求	393
7.12.7 代码示例	370	8.4.4 解决方案	393
7.12.8 相关模式	372	8.4.5 实现	394
第 8 章 并发模式	373	8.4.6 结论	396
8.1 单线程执行	374	8.4.7 .NET API 用法	397
8.1.1 概要	374	8.4.8 代码示例	397
8.1.2 背景	374	8.4.9 相关模式	399
8.1.3 要求	377	8.5 阻行 (Balking)	399
8.1.4 解决方案	377	8.5.1 概要	399
8.1.5 实现	377	8.5.2 背景	399
8.1.6 结论	378	8.5.3 要求	400

8.5.4 解决方案	401	8.8.9 相关模式	427
8.5.5 实现	401	8.9 双缓冲	427
8.5.6 结论	401	8.9.1 概要	428
8.5.7 代码示例	401	8.9.2 背景	428
8.5.8 相关模式	403	8.9.3 要求	429
8.6 调度器	403	8.9.4 解决方案	429
8.6.1 概要	403	8.9.5 实现	430
8.6.2 背景	403	8.9.6 结论	432
8.6.3 要求	405	8.9.7 .NET API 用法	432
8.6.4 解决方案	405	8.9.8 代码示例	432
8.6.5 实现	407	8.9.9 相关模式	448
8.6.6 结论	407	8.10 异步处理	449
8.6.7 代码示例	408	8.10.1 概要	449
8.6.8 相关模式	413	8.10.2 背景	449
8.7 读/写锁模式	413	8.10.3 要求	451
8.7.1 概要	413	8.10.4 解决方案	451
8.7.2 背景	413	8.10.5 实现	452
8.7.3 要求	415	8.10.6 结论	454
8.7.4 解决方案	415	8.10.7 .NET API 用法	454
8.7.5 实现	416	8.10.8 代码示例	454
8.7.6 结论	416	8.10.9 相关模式	456
8.7.7 .NET API 用法	417	8.11 Future 模式	456
8.7.8 代码示例	417	8.11.1 概要	457
8.7.9 相关模式	422	8.11.2 背景	457
8.8 生产者-消费者	422	8.11.3 要求	458
8.8.1 概要	422	8.11.4 解决方案	458
8.8.2 背景	422	8.11.5 实现	460
8.8.3 要求	423	8.11.6 结论	461
8.8.4 解决方案	423	8.11.7 .NET API 用法	462
8.8.5 实现	425	8.11.8 代码示例	463
8.8.6 结论	425	8.11.9 相关模式	468
8.8.7 .NET API 用法	425	参考文献	469
8.8.8 代码示例	425		