

Len Fenster 在开发团队中的专业知识和经验表明，他是向开发人员讲授如何最有效地使用企业库的最佳人选”



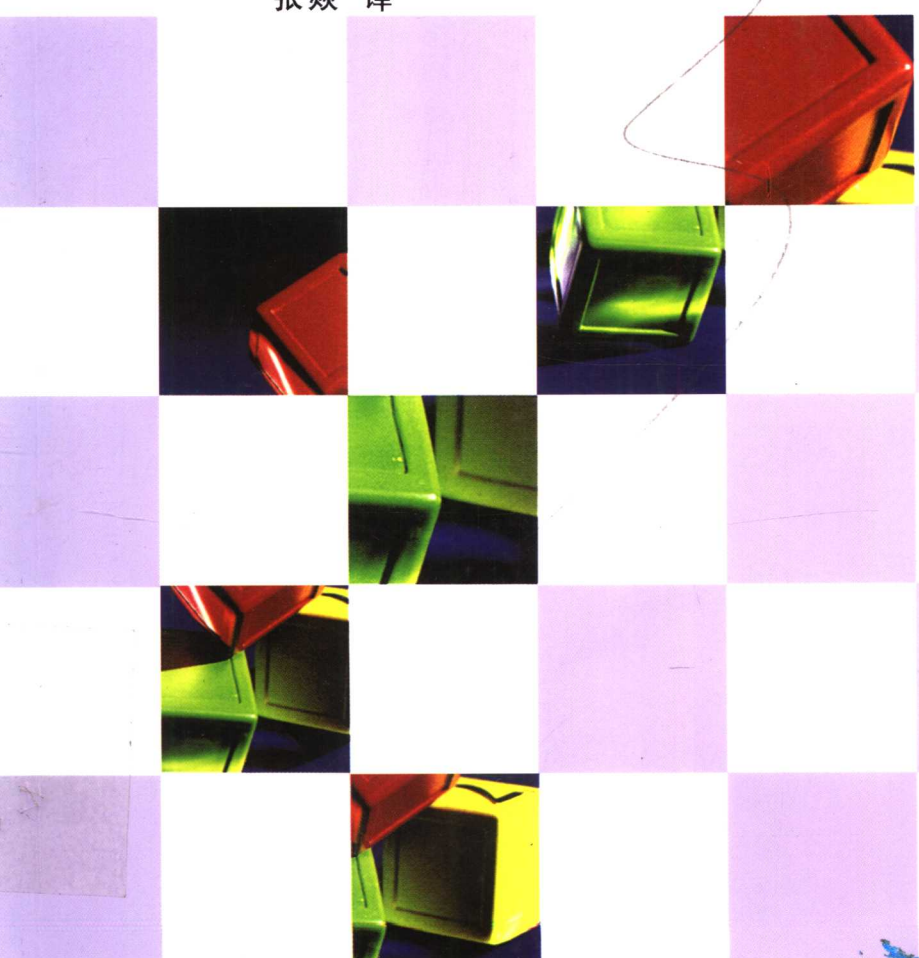
——出自 Tom Hollander 为本书写的前言

Tom Hollander 是微软公司模式与实践产品经理

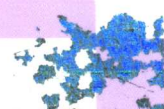
Microsoft Enterprise Library

高效开发指南

Len Fenster 著
张焱 译



Microsoft®
.net
Development
Series



清华大学出版社

TP393.09

112

2007

Microsoft Enterprise Library 高效开发指南

Len Fenster 著

张 焱 译

清华大学出版社
北 京

Simplified Chinese edition copyright © 2007 by PEARSON EDUCATION ASIA LIMITED and TSINGHUA UNIVERSITY PRESS.

Original English language title from Proprietor's edition of the Work.

Original English language title: Effective Use of Microsoft Enterprise Library by Len Fenster, Copyright © 2006

EISBN: 0-321-33421-3

All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison-Wesley.

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macao).

本书中文简体翻译版由 Addison-Wesley 授权给清华大学出版社在中国境内(不包括中国香港、澳门特别行政区)出版发行。

北京市版权局著作权合同登记号 图字: 01-2006-4347 号

版权所有, 侵权必究。侵权举报电话: 010-62782989 13501256678 13801310933

本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签, 无标签者不得销售。

图书在版编目(CIP)数据

Microsoft Enterprise Library 高效开发指南 / (美) 芬斯特 (Fenster, L.) 著; 张焱译. —北京: 清华大学出版社, 2007.2

书名原文: Effective Use of Microsoft Enterprise Library

ISBN 978-7-302-14501-1

I. M… II. ①芬… ②张… III. 计算机网络—程序设计 IV. TP393

中国版本图书馆 CIP 数据核字 (2007) 第 003797 号

责任编辑: 常晓波

责任校对: 张 剑

责任印制: 王秀菊

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

c-service@tup.tsinghua.edu.cn

社 总 机: 010-62770175

邮购热线: 010-62786544

投稿咨询: 010-62772015

客户服务: 010-62776969

印 刷 者: 北京市清华园胶印厂

装 订 者: 三河市兴旺装订有限公司

经 销: 全国新华书店

开 本: 185×230 印张: 32.75 字数: 692 千字

版 次: 2007 年 2 月第 1 版 印次: 2007 年 2 月第 1 次印刷

印 数: 1~3000

定 价: 59.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题, 请与清华大学出版社出版部联系调换。联系电话: 010-62770177 转 3103 产品编号: 022812-01

序 言

PREFACE

做一名程序员不容易。但这也是一件好事，否则这份工作很快就会变得无聊。每个项目都带来了新的需求和新的技术挑战，产品和平台版本不断演进。

在开发过程中，除了令人兴奋的事情，还有相当多这样的感受：之前已经多次写过完全相同的代码。这是因为现实世界的需求往往需要开发人员在更高的抽象水平上写出通用的代码，基础平台并不能提供这样的功能，这些代码通常用于应用程序架构或企业在线交易应用程序中的跨业务特性。

一遍一遍地编写大致相同的代码是很无聊的，同时效率低下，还会导致一致性问题和质量问题。为了解决这个问题，多年来，微软模式与实践小组编写了可重用、可扩展、可配置的源代码组件，称为**应用程序块**。我们从一个简单的数据访问帮助类开始，称之为数据访问应用程序块，然后很快转到其他方面，包括异常管理、日志、身份验证、缓存、脱机智能客户支持。这些程序块都被证明是行之有效的做法，它们设计用于满足最常见的企业应用需求，而且用于满足每一个组织或项目的特殊需求，这些程序块可以自定义并可以扩展。

.NET 开发社区已经很好地接纳了第一代应用程序块，所以大多数人都认为模式与实践小组开发的 Enterprise Library（企业库）开创了基于代码的指导原则的新时代。企业库包含的功能和以前发布的独立程序块很相似，但在内部程序块之间取得了更强的一致性；易于扩展以支持新的需求；通过更简单的 API、文档和实例，使用起来更方便；应用中常把程序块集成起来使用。

除了交付件自身的变化，企业库另一个大的改进在于一个社区的形成——成千上万的开发人员、测试人员和架构师以及模式与实践小组通过我们的 GotDotNet、博客、Webcasts 和活动来共享问题、解决方案、经验和扩展。通过经验反馈和相互启发，已经极大地提升了企业库的价值，这是模式与实践小组自身所不能企及的。

虽然 Lenny Fenster 是微软的员工，但模式与实践小组也是通过这个公共社区知道他



的。Lenny 的角色是每天与客户在实际项目中一起工作，他有机会在项目中使用企业库，以及了解客户的新挑战，他们不是针对企业库的，而是寻求更简单的解决方案。基于这些认识，Lenny 创建了数据映射应用程序块，作为对企业库的扩展，这个团体和本书的读者都可以享用它！

在为即将发布的.NET 框架 2.0 升级企业库时，模式与实践小组非常幸运地能够把 Lenny 从他的客户工作中“借调”出来，使之成为企业库开发团队的核心成员。不仅模式与实践小组可以从他最近的客户经验中获益，也使 Lenny 获得了创建模式与实践交付文件的第一手经验。本书重点放在.NET 框架 1.1 企业库上，大多数概念对.NET 框架 2.0 版本也是适用的。Lenny 在开发团队中的专业知识和经验表明他是向开发人员讲授如何最有效地使用企业库的最佳人选。

感谢 Lenny 以及所有的人，感谢你们对微软模式与实践小组以及.NET 开发社区的支持。

—Tom Hollander, 产品经理

微软模式与实践小组

<http://msdn.microsoft.com/practices>

<http://blogs.msdn.com/tomholl>

前言

FOREWORD

CIO、CTO、架构师和开发人员都热切地期望开发出适合于许多企业的应用程序，因为这通常需要许多资源，包括时间和金钱，才能开发出核心的基本服务来支持这些应用程序。微软为开发这些支持它们的服务和代码提供了指导。本书填补了如何结合使用所有应用块的空白。这样，架构师和开发人员就可以基于微软推荐的最佳实践，以一种容易的和可重复的方式来创建可靠的面向服务的架构（SOA）。

你可能会问，怎么办？接着读下去就能得到答案。

今天许多公司都不仅面临如何尽可能地利用 Microsoft .NET 的特性和功能来创建健壮的应用程序的挑战，还面临着把这些应用程序有效组织起来的挑战，这样他们才能从面向业务的架构中获益，这种架构宣称能够同时应付现在和未来。当初次面对为某一 CxO 设计企业级应用的机会时，想象一下用一堆积木帮助你的孩子搭建一个 Sphinx 模型，就会对架构有一个很好的认识。

微软会指导你如何构建企业级的应用，以充分利用 .NET 的特性和功能。它以 Prescriptive Architecture Guidance (PAG) 文档的形式提供帮助。PAG 文档详述了架构师设计应用程序时应当考虑的不同层、组件和服务。把 PAG 看成是 Lego 积木箱里的 Sphinx。微软也以 .NET 应用程序块的形式提供了 Lego 块。一个应用程序块是实现组成应用程序的一个层上的一个组件或服务。

那么，有什么问题？

问题是没有使用 Lego 块创建应用程序块的指令手册或文档化的流程。从哪一个块开始？是不是要用到所有的块？盒子里是不是漏掉了什么块？如何把它们最好地组合在一起？

本书就充当了这样一个手册，它指明如何用微软提供的程序块创建箱子上的图画。本书会帮助你在需要时重塑一些应用程序块，并明白如何创建新的程序块。它还介绍了微软提供的帮助你把程序块组织在一起的工具，以及围绕面向业务的架构组合 .NET 应

用程序块的新的需求。

应用程序块简史

重用软件资产的好处已经被吹捧了好多年。今天，开发人员花费了大量的时间和精力来寻找他们需要的软件或代码库以避免“重新发明车轮”。开发常常带着“这可能以前有人做过”的想法在互联网上查找代码示例或可重用的软件库。有时能找到解决方案，有时不能。但是，即使能找到解决常见应用程序问题的软件库，使用它们也是一个挑战。资源的设计和質量以及资源的未来方向都必须进行评估。资源所依赖的附属物也要评估以决定使用它是否会导致企业偏离他们的战略路线。

微软应用程序块试图通过提供一个大多数企业应用都需要的核心资产库进行帮助。这始于 2002 年 4 月最初引入的数据访问应用程序块。这个程序块之所以流行是因为它简单易懂（它的接口非常简单，没有依赖关系），同时它涵盖了大多数项目访问微软 SQL Server 数据库所需要的常见操作。

在接下来的几年里，又引入了更多的应用程序块；但是，对它们的接纳程度并不相同。总体来讲，这些应用程序块都可以被认为是成功的，但当有了越来越多的程序块时，它们的使用就产生了问题。这些问题都围绕着以下事项。

- **一致性。**应用程序块是微软在几年间由不同的团队创建的，它们之间并没有一致的原则。结果就是应用程序块在方法和包装上大不相同。如果使用一个新的应用程序块，用户发现很难理解。另外，因为程序块的方法不一致，许多用户发现很难在同一个项目让它们协同工作。
- **配置。**大多数应用程序块依赖于复杂的配置，几乎没有工具支持或验证来帮助正确地配置应用程序。配置应用程序块必须手动创建并修改 XML 文件和测试运行时的结果。这是大多数最初的程序块用户面临的普遍问题之一。
- **复杂性。**数据访问应用程序块之后的许多应用程序块都难于理解。但是，为了有效使用应用块，开发人员需要通过理解如何配置、开发和扩展程序块来快速磨合。缺少示例和指南来帮助开发人员快速掌握一个应用程序块。另外，程序块提供的文档并没有为大多数开发人员理解如何使用程序块提供足够的信息。
- **依赖关系。**公司经常想用一個程序块，但它们发现不想接纳这个应用块的依赖关系。一个应用程序块经常与其他应用程序块是紧密相关的。例如，许多程序块都依赖于异常管理应用程序块。但是，如果一个企业不想使用异常管理应用程序块，那么它要么手动去掉依赖关系，要么根本不用这个程序块。同时，最

初的日志应用程序块依赖于 Enterprise Instrumentation Framework (EIF)。不想使用 EIF 的用户也不能使用日志程序块。

企业库是应用程序块的演进。它不是微软的产品，因此你也不必为此付费。另外，企业库的所有代码都是公开的。这都与微软的其他产品有很大不同。企业库有 7 个应用程序块——配置、数据访问、缓存、异常处理、日志和检测、安全和加密，这个库真正能够应对企业应用中遇到的挑战。它不是 .NET 框架的一部分，但它确实是为了填充 .NET 框架中的“空白”，直到这时这个框架才吸纳了这些功能。

企业库的远景是吸取最初的应用程序块的教训，不但要创建一套大多数企业应用所需要的最基本的应用程序块，还要创建一个完整的生态系统。在这里，微软客户、合作伙伴、团体成员和模式与实践小组都能提供可重用的资源，根据企业的需要，这些资源可以合并到代码库中。企业库背后的核心思想由四条原则定义，它们指导着企业库中所有应用程序块的开发。

- **一致性。**每一个应用程序块都应该采用一致的设计模式和实现方法。这样，如果理解了一个程序块，再理解和使用其他程序块都会很容易，并能很快地提高生产率。
- **可扩展性。**应用程序块必须定义可扩展点，通过插入自己的代码，开发人员可以自定义程序块的行为。目标是预测扩展的需要，最小化自定义代码时对核心元素的修改。通过设计可扩展的应用程序块，应用程序块能够适用于不同的环境，有更多机会适应企业，如果不这样设计，就很难达到目的。
- **易用性。**对于开发人员，应用程序块必须易用。因此，企业库团队提供了：
 - 企业库配置工具。**因为配置应用程序块被证明最难使用，所以创建了图形化的配置工具来改善这方面的体验。配置工具可以使创建、修改、删除和验证所有企业库自带应用程序块的配置信息变得很容易。这就不必手动修改 XML 文件让应用程序块正确工作。
 - 更简单的安装过程。**建立一个开发环境来使用任一个应用程序块都变得很容易。
 - 清楚的文档和示例。**每一个应用程序块的文档都包括如何使用应用程序块的关键场景，并说明了如何执行最常用的任务。
- **完整性。**应用程序块应设计成能在一起工作，并通过测试保证能这样做。但是，也应该有可能使用其中一个应用程序块而不依赖于其他的程序块。贯穿企业库，应用程序块的整合都是以松耦合的目标进行设计的。依赖于其他程序块的应用程序块完全可以用另外的资源替换掉，而不影响调用块。

最后，有重要的一点需要指出，企业库意在以具体到代码的方式提供架构指导。虽

然它提供了全部源代码，允许每个用户修改它，但是在修改代码前要小心，因为每个应用块都有意在推行应对企业级应用常见的特定挑战的最佳实践。事实上，本书的许多章节都对应用程序块的设计和微软模式与实践小组发布的指南进行了比较。对于许多应用程序块，它们的实现就是遵循特定程序块蓝图指南的直接结果。

例如，数据访问应用程序块的设计重点关注如何在应用程序的数据层使用数据库提供程序。它的指南可以在 *Application Architecture for .NET: Designing Applications and Services*¹ 出版物中找到。缓存应用程序块的实现指南在 *Caching Architecture Guide for .NET Framework Application*² 中进行了描述。我创建了数据映射应用程序块，在第 9 章提供了更多的细节，附录 A 意在实现在分布式应用中创建和使用数据访问逻辑组件的指导原则，这也在 *Architecture for .NET: Designing Applications and Services* 出版物中得到了体现。

关于本书

.NET 开发人员和架构师可能会对本书感兴趣，因为它集中讲述了如何利用企业库应用程序块和微软建议的最佳实践创建企业应用。我假设读者有 Visual Basic.NET 或 C# 方面的经验。在每一处我认为有必要指出如何使用一个应用程序块或说明程序块如何工作的地方，都包含了两种语言的代码示例。

我相信大多数读者都可分为两类：一类是那些想知道如何设计企业库应用程序块以及如何利用这种设计的优点扩展企业库以更好地满足他们需求的人，另一类是那些只想更多了解关于如何用企业库配置和开发应用程序方面信息的人，他们不想扩展任何应用程序块，也不想创建他们自己的应用程序块。

本书试图同时照顾到这两类读者。每一章都针对企业库中一个不同的应用程序块或者企业库中某个特定的重要方面（例如，配置设计时的属性，或如何创建一个自定义的应用程序块）。对于每一个应用程序块，我都在开头部分详细解释这个程序块是如何设计的，即应用程序块中可用的扩展点，并至少举一个自定义的真实例子来说明如何创建你自己的应用程序块扩展。每一章的结尾对如何配置应用程序块以及如何充分利用程序块提供的优点进行了简单说明。

如果你对一个程序块的设计和扩展不关心，可以直接跳到该章后面的部分。如果你已经基本知道了如何用程序块配置和开发一个应用程序，但是想知道如何扩展程序块来满足你的需求，只需阅读第一部分。如果想尽可能多地了解一个程序块，就要从头到尾通读这一章。

1 在 <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/distapp.asp> 上可以找到。

2 在 <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/CachingArch.asp> 上可以找到。

本书由以下 9 章和 3 个附录组成。

第 1 章：配置应用程序块运行时

本章之所以重要有几大原因。它讲述了配置应用程序块是如何设计的，企业应用可以用它方便地读写配置数据，而完全与实际的配置数据的存储类型和位置没有关系。但是，更重要的是，企业库中所有其他的应用程序块都依赖于配置应用程序块。配置应用程序块是唯一依赖于其他程序块的应用程序块。因为所有的应用程序块都由配置驱动，理解这个程序块在为其他程序块提供功能中的角色至关重要。本书中许多现实的企业库扩展都依赖于对配置运行时的最基本的理解。

第 2 章：配置应用程序块设计时

第 1 章之所以重要的理由同样适用于本章。企业库主要的设计目标之一就是更容易配置应用块。配置应用程序块的设计时特性提供了达成这个目标的功能。在为任何一个应用程序块设计扩展或创建一个新的应用程序块时，为了使新的扩展易于使用和配置，理解配置应用程序块的设计时特性是如何工作的非常重要。

第 3 章：数据访问应用程序块

数据访问应用程序块提供了一套类和接口，它们封装了微软推荐的设计数据层的许多最佳实践。确切地讲，企业库数据访问应用程序块提供的解决方案能够满足用数据库提供程序来生成更清洁、更可管理的代码的需求。这个版本的数据访问应用程序块另一个新的设计目标是提高使用这个程序块的代码在不同类型数据库之间移植的可能性。

第 4 章：缓存应用程序块

企业库缓存应用程序块是对 *Caching Architecture Guide for .NET Framework Applications* 一书所倡导的缓存解决方案蓝图的实现。因此，它提供了一个简单而且一致的编程接口，在编写应用程序代码时与所用的缓存存储类型没有多大关系。这样，即使缓存数据保存的位置变了，也不会影响到应用程序的代码。

第 5 章：异常处理应用程序块

企业库的异常处理应用程序块是一个策略驱动的系统，由管理员决定在应用程序中如何处理异常，而管理人员则重点关注为什么以及在哪里处理异常。这个程序块是基于微软模式与实践小组所倡导的最佳实践而实现的。这个应用块提供了一个非常简单的编程接口，应用程序开发人员无需关心异常是如何处理的。这样，即使异常处理的方式变了，或者为特定的场景增加或删除了异常处理器，应用程序的代码也不会受到影响。

第 6 章：日志和检测应用程序块

日志和检测应用程序块提供了一种标准并且一致的记录和处理消息的方法，不管消

息的目的地是事件日志、文件、数据库、MSMQ 还是 WMI 事件。这个应用程序块使得开发人员很容易在应用中引入日志和跟踪功能，而无需知道什么时间、在什么地方，甚至是否进行了记录。如果已经提供的功能不能满足你的需求，这一章将介绍如何扩展日志和检测应用程序块以提供把信息记录到其他数据源的能力。

第 7 章：安全应用程序块

安全应用程序块的主要设计目标是为了便于企业向它们的应用中添加身份验证、认证、角色管理、安全缓存，以及配置文件管理功能。安全应用程序块自带相关的安全数据库、安全数据库管理平台 and 数据库的安全提供程序，这样可以更容易地添加安全功能，而不需要依赖外部的用户存储，例如，微软活动目录（Microsoft Active Directory）。同时，许多企业已经有了适当的系统来存储它们的用户信息，安全应用程序块很容易扩展以满足这些企业的需求。这一章会告诉你如何做。

第 8 章：加密应用程序块

加密应用程序块可以使开发人员更容易地给他们的应用程序增加功能，如：加密或解密数据、创建和比较哈希表。加密应用程序块用到了 .NET 加密 API 的许多资源，同时避免了由此带来的复杂性。

第 9 章：创建应用程序块

到目前为止各章的内容都是关于如何扩展应用程序块以满足特定企业需求的。但是，企业库的可扩展性并不仅限于扩展每一个应用程序块。企业库团队实际上期望能够创建新的应用程序块来解决特定的企业需求。企业库自带的应用程序块可以解决大部分企业应用面临的挑战性问题。如果企业库自带的应用程序块不能解决重复出现的挑战，那么最好创建新的应用程序块来解决这个需求，并且和企业库自带的应用程序块一视同仁。在这一章我会带你一同领略创建应用程序块的各个步骤来解决你的问题。

附录 A：数据映射应用程序块

第 9 章描述了创建新的应用程序块的步骤。这个新的应用程序块——数据映射应用程序块是要让开发人员创建数据访问逻辑模块变得更容易，这是微软模式与实践小组推荐的最佳实践。本章将提供这个应用程序块的设计细节并且告诉你如何用它配置和开发一个应用程序。

附录 B：创建 .NET 托管数据提供程序

第 3 章描述了企业库自带的数据库提供程序，并说明了如何通过创建一个新的数据库提供程序扩展数据访问应用程序块。数据库提供程序的一个前提条件是要有一个 .NET 托管数据提供程序，它要实现需要被数据库提供程序返回的接口。在这个附录



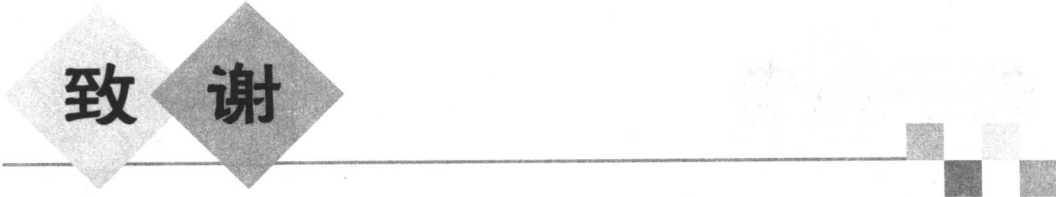
中,说明了如何创建一个.NET 托管数据提供程序,只有在极少数情况下你才会用到它。

附录 C: .NET 框架 2.0 企业库

本书解释了.NET 框架 1.1 企业库的设计,以及如何扩展、开发和配置企业库应用程序块。2006 年 1 月,微软发布了.NET 框架 2.0 企业库。本附录将重点介绍二者之间重大的不同。

参加本书翻译的人员有:张焱、陈宗斌、蔡京平、李毅、毕蓉蓉、祁海生、张贺乾、史宁、刘绿生、孙雷、蔡加双、安东辉、米翔娟、刘颜、王宇宇、沈程亮、陆晓萍、金国良、俞群、李正智、赵敏、陈征、陈红霞、张景友、易小丽、陈婷、管学岗、王新彦、金惠敏、张海峰、徐晔、戴锋、张德福、张士华、张锁玲、杜明宗、高玉琢、王涛、刘晓捷、董礼、何永利。

致 谢



首先我要感谢本书的技术评审——Bill Ryan、Brian Davis、Klaus H. Probst 和 Les Smith，你们花费了大量精力审阅本书的原稿，并给出了非常有价值的反馈。你们的建议对于达到本书的目标并让风格适合于目标读者非常有帮助。

特别感谢 Scott Densmore 和 Tom Hollander。你们严格而又公正的审阅对于把本书打造成现在的模样至关重要。你们指出了需要进一步讨论的重要内容，并毫不留情地告诉我哪些看起来不太正确的内容应该删除。没有你们的真知灼见，这本书不可能以今天的形式出版。

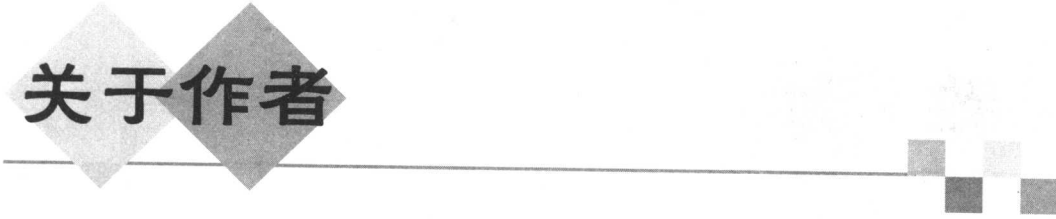
各位编辑（Joan Murray、Jessica Damico、Jim Markham 和 Rebecca Greenberg）指导我完成了本书的写作，你们的建议是无价之宝。在这里我积累了很多写书方面的经验。当前景不妙时，你们从来都没有对我丧失信心，并一直鼓励我。谢谢你们！

Amber，是你每个晚上都陪伴在我的周围，甚至到凌晨三点钟。你是我的好狗狗。

Lexa、Evan 和 Ana，我尽量不因为写这本书而剥夺我们在一起的时间，但是我知道有时这是很困难的。当爸爸不得不写作时，谢谢你们一贯的理解，我为有时通宵写作到凌晨向你们表示歉意。你们的爱让我能够不知疲倦地写作。

Lisa，谢谢你的爱心和耐心。没有你的支持我不可能完成这件事。我们错过了很多夜晚、周末、节假日和假期才完成了本书的写作。感谢你持续的鼓励和每次还差一点才能写完时表现出来的耐心。我期望弥补一年来写作本书错过的美好时光。我爱你。

关于作者



Len Fenster 是微软咨询服务美国东部地区 .NET 开发的首席架构师。他在微软的八年时间，工作重点是帮助企业创建基于微软技术的健壮的应用程序。最近 Len 和微软模式与实践小组一起工作致力于下一个版本的企业库的开发。甚至在进入微软之前，他就在领导一个全球性开发人员和架构师团队创建基于微软技术的分布式应用。自从加入到 .NET 后，他就作为微软咨询服务的企业架构师利用他丰富的经验帮助许多企业把 .NET 融入到他们自己的技术策略中去。Len 定期向很多公司以及在架构论坛上讲述构建基于 .NET 和面向业务的架构。

目 录

CONTENTS

第 1 章 配置应用程序块运行时 1

- 什么是配置应用程序块 1
- 配置应用程序块运行时的设计 2
 - 读写配置数据 2
 - 与物理存储位置无关 6
 - 企业库中的提供程序 8
 - 管理配置信息 36
- 使用配置应用程序块进行开发 42
 - 读配置数据 42
 - 写配置数据 43
 - 检测变更通知 43
 - 清除缓存的配置数据 45
 - 使用 ConfigurationContext 访问配置数据 45
- 小结 47

第 2 章 配置应用程序块设计时 48

- 配置和企业库配置工具 48
 - 创建配置段 50
 - 配置 StorageProvider 50
 - 配置 Transformer 59
 - 对配置数据进行加密 61
- 配置设计时背后的技术细节 64
 - 使用 System.ComponentModel 命名空间 64

配置体系	69
配置菜单项和命令	69
ConfigurationDesignManager 类	70
小结	78

第 3 章 数据访问应用程序块 79

什么是数据访问应用程序块	79
数据访问应用程序块的设计	81
数据库提供程序	81
数据库命令封装器(Wrapper)	89
参数处理和 ParameterCache	91
DatabaseFactory 和 DatabaseProviderFactory 类	93
创建自定义数据库提供程序	95
使用数据访问应用程序块的 API	101
配置数据库提供程序	101
使用数据访问应用程序块进行开发	106
小结	129

第 4 章 缓存应用程序块 130

什么是缓存应用程序块	131
性能	131
扩展性	131
可用性	132
为什么不用 ASP.NET 缓存	132
缓存应用程序块的前期版本	133
缓存应用程序块的设计	133
CacheManager, CacheManagerFactory 和 CacheFactory	134
缓存对象	138
CacheService 对象	139
CacheStorage	147
用缓存应用程序块进行开发	162
配置缓存应用程序块	162
使用缓存应用程序块的 API	167
小结	176

第 5 章 异常处理应用程序块 177

- 什么是异常处理应用程序块 178
 - 与异常管理应用程序块的关系 178
 - 自定义异常和异常层次结构 179
- 异常处理应用程序块的设计 181
 - ExceptionPolicy 类 181
 - 异常处理器 186
- 用异常处理应用程序块开发 215
 - 配置异常处理应用程序块 215
 - 使用异常处理应用程序块的 API 223
- 小结 225

第 6 章 日志和检测应用程序块 226

- 什么是日志和检测应用程序块 227
 - 与日志应用程序块和 EIF 以前版本的关系 227
- 日志和检测应用程序块的设计 228
 - 日志创建 228
 - 筛选 243
 - 日志分发 246
 - 路由 254
 - 跟踪 271
- 开发日志和检测应用程序块 280
 - 配置日志和检测应用程序块 280
 - 使用日志和检测应用程序块的 API 289
- 小结 292

第 7 章 安全应用程序块 294

- 什么是安全应用程序块 295
 - 与授权和配置文件应用程序块的关系 295
- 安全应用程序块的设计 296
 - 安全数据库 296
 - 认证 297
 - 缓存安全令牌 304
 - 授权 310