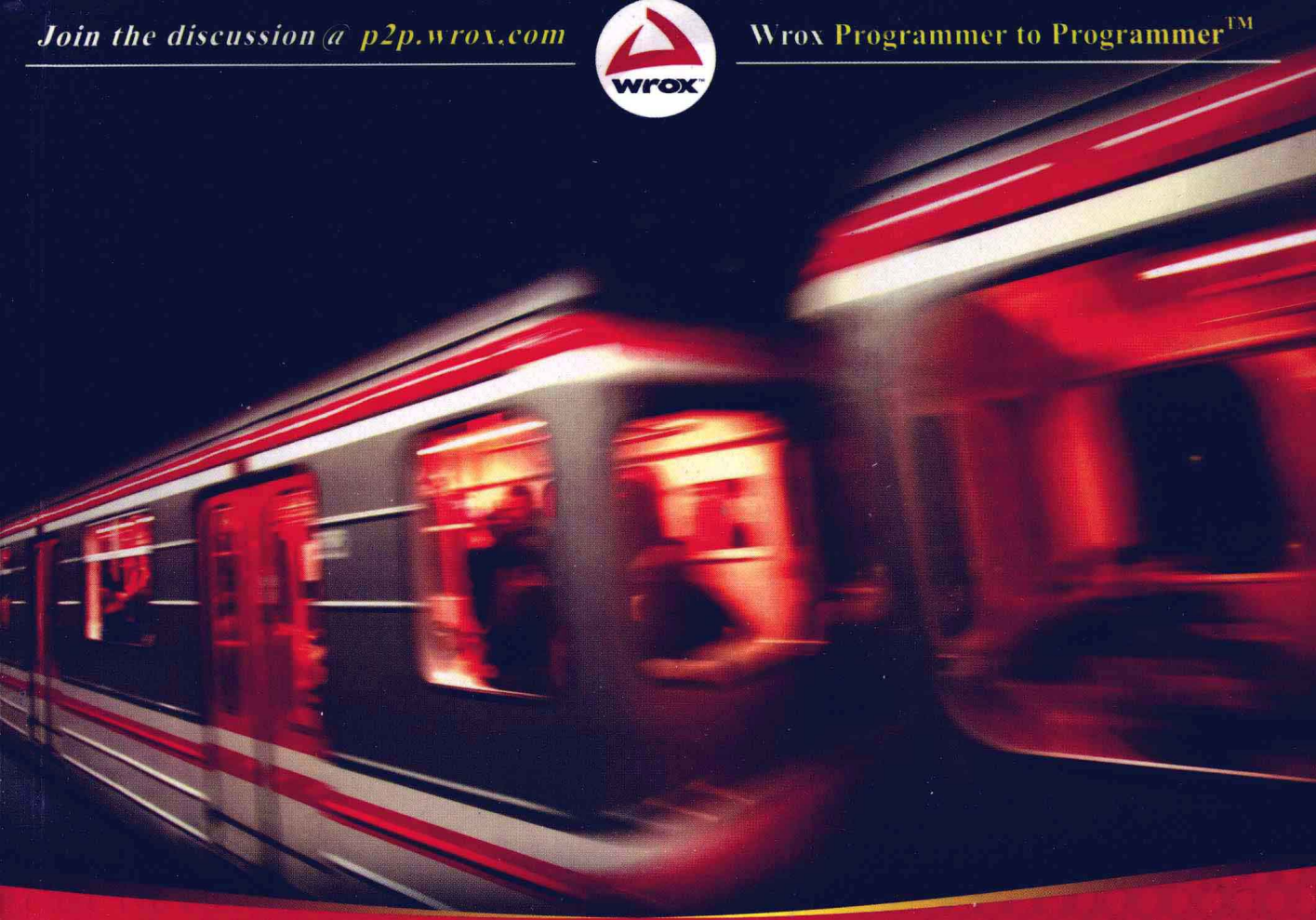

Professional Enterprise .NET

# 精通.NET企业项目开发：最新的模式、工具与方法

(美) Jon Arking 著
Scott Millett
田尊华 译

清华大学出版社

# 精通.NET 企业项目开发：

## 最新的模式、工具与方法

(美) Jon Arking
Scott Millett 著

田尊华 译

清华大学出版社

北 京

Jon Arking Scott Millett
Professional Enterprise .NET
EISBN：978-0-470-44761-1


**图书在版编目(CIP)数据**

精通 NET 企业项目开发：最新的模式、工具与方法/(美) 阿金(Arking，J)，(美) 米勒(Millett，S) 著；田尊华 译. —北京：清华大学出版社，2011 3

书名原文：Professional Enterprise NET

ISBN 978-7-302-25024-1

Ⅰ 精… Ⅱ. ①阿… ②米… ③田… Ⅲ 主页制作—程序设计 Ⅳ. TP393 092

中国版本图书馆 CIP 数据核字(2011)第 028260 号

**责任编辑**：王 军 张立浩
**装帧设计**：孔祥丰
**责任校对**：胡雁翎
**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
**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**：北京市世界知识印刷厂
**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**：29.75 **字 数**：724 千字
**版 次**：2011 年 3 月第 1 版 **印 次**：2011 年 3 月第1 次印刷
**印 数**：1～3000
**定 价**：68.00 元

产品编号：035683-01

# 译 者 序

一般的程序设计与企业开发是根本不同的，这就好比一名围棋新手与一名围棋高手下棋一样。一般的程序设计只需要掌握编程语言的语法就可以编写程序，同样，围棋新手只懂基本规则就能下棋；然而，企业开发首先要考虑的是架构和模式，这就好比深谙各种套路的围棋高手下围棋时的初始布局。围棋布局不好，那么即便是在局部具有优势，但是随着棋局的进一步深入发展，就很难在全局上占据优势，由此可能导致满盘皆输。企业开发看重的正是这种布局，它不期望很快获得结果，但是却能够让获得的结果具有良好的可测试性、可维护性和可扩展性。这是一般的程序设计人员难以企及的目标。

如果希望由一般的程序设计人员过渡到企业系统设计师，那么本书正是您的最佳向导。本书对企业系统开发与设计的基本准则、开发方法和设计模式进行了深入和具体的讲解。这些是进行企业开发所必备的知识，它们能够引导开发人员快速地过渡到企业开发。企业开发不仅要牢记诸如可测试性、可维护性和可扩展性这样的基本准则，同时还要熟练掌握达成这些核心价值所需的开发方法和设计模式。对于所有这些内容，本书都进行了深入、细致的讲述。

本书具有的显著特征是目标明确，就是帮助有志于企业开发的人员由非企业程序设计顺利地过渡到企业设计。为此，要求读者具有一定的程序设计经验，并准备付出大量时间和精力来学习和实践众多的企业系统开发方法和设计模式。为了帮助读者快速过渡到企业开发，本书首先从企业开发的基本准则和核心价值进行了深入细致的讨论，然后结合按揭贷款申请系统的构建，循序渐进地讨论了企业开发的各种方法与设计模式。本书虽然涉及很多概念，但由于穿插着大量的代码示例，从而使得这些概念并不显得抽象，而是具体的和可操作的。读者如果能够按照书中的讲解和示例代码逐步构建一个完整的按揭贷款申请系统，那么获得的效果应该是最佳的。

本书由国防科学技术大学的田尊华翻译。他长期从事计算机科学中大型系统的研究与开发工作，具有丰富的程序设计经验和大型系统开发经验。肖国尊组织整个翻译过程，负责全书的质量和进度控制，为本书的出版做了大量的协调工作。

企业开发涉及的内容非常丰富，而且不断出现新技术、新方法和新工具，加之译者的水平和学识有限，译文中翻译的不妥之处在所难免，恳请读者批评指正。读者反馈可以发送到 wkservice@vip.163.com。

# 作 者 简 介

Jon Arking 是一位企业软件架构师，其工作地点大部分时间都在费城。到目前为止，他设计、开发和管理多层系统已经超过了 14 年之久，并专门从事系统移植和分布式体系结构的设计。Jon 具有多种语言和平台的编程经验，他在其职业生涯中花费了很多时间进行设计系统、管理团队、教授课程、演讲和访谈，并发表了各种技术主题的文章。他的 Arking 技术公司则专注于为费城范围内的大型公司设计企业系统。

Scott Millett 生活在英格兰南部的朴茨茅斯市的 Southsea，他是 Wiggle.co.uk 的一位资深开发人员，Wiggle.co.uk 是一家电子商务公司，专注于英国范围内的自行车和游艇健身运动。自从.NET 1.0 版开始，他就开始基于.NET 进行开发，并获得了 Microsoft 认证专家 Web 开发证书。他是 ASP.NET 论坛的长期贡献者，在编写.NET 和基于.NET 工作之余，他经常会在阿灵顿休闲和享受音乐，并且参加夏季在英国举办的所有主要的音乐节。如果希望与 Scott 讨论本书、任何关于.NET 的内容或英国音乐节场景，那么请给他写邮件，他的邮件地址是 scott@elbandit.co.uk；或者访问他的博客 www.elbandit.co.uk/blog。

# 内 容 提 要

本书是专门针对有兴趣学习最新企业开发方法的微软程序员而编写的权威指导书籍。本书全面深入地介绍了企业系统开发中涉及的体系结构设计方法和各种相关的设计模式，尤其是对最新流行的各种设计模式进行了详细介绍，包括纵向的来龙去脉和横向的优缺点比较。虽然各章之间都是相互独立的，不需要读者预先阅读前面的所有章节，但又以按揭贷款申请的例子为主线，由无到有、由浅入深地将企业系统构建相关的各种核心要素串联在一起，从而使讲解过程连贯有序。

本书的主要内容分为 4 大部分：第 I 部分是第 1 章和第 2 章，主要介绍了企业开发的总体概念，包括企业体系结构、企业开发准则、Microsoft 企业开发的历史与现状，以及企业代码编写方式(包括模块化、松散耦合、依赖倒置和测试驱动)；第 II 部分是第 3~5 章，结合代码示例，深入详细地讲解了封装类、测试驱动开发和依赖倒置；第III部分是第 6~12 章，本部分属于核心内容，结合按揭贷款申请的示例，详细讲解了企业开发涉及的各种设计模式，这些设计模式都是为了达成企业系统的特定目标；最后是附录部分，主要是为不太熟悉.NET 平台的读者准备的，介绍了 C# .NET 的基础知识。

本书专门针对具有一定微软应用程序开发背景的编程人员，尤其适合于具有 C#和 ASP.NET 开发经验的人员。为此，阅读本书要求读者至少熟悉一种 Microsoft .NET 所支持的开发语言，并具备一定的程序设计经验。

# 前　言

计算机编程与开发业务软件并不相同。很多人认为，不管如何编程以及为谁编程，编程本身都是一样的，两者都不能脱离实际。确实如此，为了完成别出心裁且需要付出大量辛劳的任务，必须具有编程技能，但这只不过是起码的要求而已。目前，针对特定的业务创建软件要求具有大量不同语言和学科方面的知识，同时要求具有低层的编程技能和高层的综合设计经验。但首要的要求是，必须具有耐心，并且能够接受新思维。

当然，很多人并不认同这个观点。大多数人都知道，总是存在那么一、两个程序员，他们在工作时总是尽可能地将自己关在一个独立空间里，不接触业务领域的复杂性。有时候这样做有一定的好处。毕竟，并不是每个人都需要创建优秀的软件而使他们的整个公司都高奏凯歌。小型应用软件开发在业界总存在其立足之地。然而，如果要将其中一些应用程序发展为更大规模的企业级系统，那么就一定会遇到麻烦。小型应用软件开发方法往往能够在短期内交付令人满意的结果，然而当需要回过头来扩展该软件的使用范围和功能集时可能就很困难。因此，当软件开发发展到公司范围内的企业级系统时，企业级设计的知识就会大有帮助。

为了获得设计良好的系统，相关的努力并不鲜见。从计算机时代到来开始，IT 专家们就一直争辩到底该如何尽可能地在快速的应用程序开发与正确的软件设计之间进行权衡。这是最佳的设计意图与所面临的、异常严格的交付时间之间真正的博弈。然而，随着业务变得与 IT 更加紧密相关，IT 专家们开始在企业级设计上投入精力，希望前期的投入可以节省后期阶段的投入。尽管这种可能性是完全存在的，但是开发人员需要知道如何正确地应用企业模式(enterprise pattern)，并以适合他们客户的方式来应用企业级设计。

Microsoft 开发人员熟悉这种争论。平台针对的目标几乎总是特别关注快速的应用程序开发，Microsoft 应用程序在很长的一段时间内都被认为设计很差，为了加快进入市场的步伐，他们往往牺牲了应用程序的可扩展性与灵活性。企业方法论(enterprise methodology)越来越被人们所接受，新一代的 Microsoft 开发人员们发现他们正面对着令人生畏的任务——学习这些新模式，并将它们加入现有的技能集和用于应用程序开发。但是，在建立良好设计的应用程序与快速交付的高要求之间存在差距，这种差距困扰了 Microsoft 许多年。有如此之多的企业模式需要学习，Microsoft 程序员需要实用资源的帮助，以指导他们逐一掌握这些企业模式，并帮助他们理解更新自身代码和编程经验的最佳途径。

本书是一本针对有兴趣学习最新企业开发方法的 Microsoft 程序员的权威指导。本书的目标是为开发人员讲述不同的模式与方法，这些模式和方法有助于使他们的代码更加清晰和更加易于维护。对于中级和高级 Microsoft 程序员，如果他们寻求将应用程序和自身技能集转移到更新和更灵活的企业方法上来，那么本书将竭诚为他们提供相关的路线图。

## 本书涵盖的范围

本书旨在介绍一些越来越流行的软件开发模式和方法。本书专门针对那些具有一定Microsoft 应用程序开发背景的编程人员，尤其适合于具有 C#和 ASP.NET 开发经验的人员。本书并不打算成为一本针对所有与企业相关的包罗万象的权威资源。企业设计是一个非常宽泛的主题，涵盖了相关书籍和资源所涉及到所有主题。本书并不是针对所涉及的大量宽泛主题进行综合介绍。对于测试驱动开发(Test Driven Development，TDD)、中间件(Middleware)设计模式或基于 ASP.NET 的 Web MVC 这样的主题，如果读者希望寻求深入的了解，那么最好寻找专门针对这些主题的书籍。在开始讨论企业概念以及介绍可测试代码方法和设计方法时，您首先应该熟悉以松散耦合的、可测试的方式组合代码的一些不同方法。后续章节逐步介绍了包含这些方法的一些工具，例如 Spring.Net、NHibernate 和 ASP.NET MVC。这些主题的结合有助于使开发人员弄清楚这些不同的模式在一起工作的方式，并最终帮助他们决定哪些部分是最适合他们的业务的。尽管本书确实讲述了一些原有的技术，但其意图主要是为企业开发人员提供有益的参考。本书中的各章相对独立，不需要读者预先阅读前面的所有章节。为了便于记忆，下面给出了每章的内容大纲。

**第 1 章：企业设计概念**

该章首先讨论一些与企业开发相关的核心概念。这些概念与实际的编码并没有太大关系，而与企业设计背后的思想，以及为什么要接受企业设计更为相关。从企业体系结构的概念出发，着重定义了企业体系结构的含义以及适用于什么样的对象。然后，开始探讨企业开发的概念，主要焦点在于企业开发的核心价值，如可靠性(reliable)、可维护性(maintainability)和关注点分离(separation of concern)。随后，该章将重点转移到讲述软件设计的历史，使读者了解企业设计核心价值的演进历程。在该章最后，讨论了一些逐步被企业开发人员所接受的流行工具。

**第 2 章：企业代码**

第 2 章介绍了与企业开发有关的一些新的编码概念。详细地解释了像模块性(modularity)和松散耦合(loose coupling)这样的概念，同时用一些简单的代码示例帮助说明这些概念。单元测试(unit testing)是企业体系结构中的一个强大驱动力，在模块性和松散耦合的背景下，仔细地解释了单元测试的概念。在此还介绍了控制反转(Inversion of Control)的概念，解释了在可测试代码库内如何使对象实例的创建可管理。在该章的最后讨论了一些可以获取的工具，利用像 NUnit、Resharper 和 Spring.Net 这样的工具有助于辅助企业编码。

**第 3 章：改变类的依赖**

第 3 章集中讨论了依赖倒置(Dependency Inversion，DI)的概念。该原理倒置了高级和低级模块之间的关系，降低了它们对具体实现的依赖性。该章最后按照依赖注入(Dependency Injection)的形式，讲述了 DI 准则的应用，依赖注入就是将低层对象(low-level object)注入到较高层对象(higher-level object)的过程。

**第 4 章：测试驱动开发**

第 4 章介绍了测试驱动开发(Test Driven Development，TDD)方法，该开发模式利用了

松散耦合和高度可测的代码设计。基于一个综合示例，将 TDD 用于说明如何通过单元测试来驱动应用程序的设计。该章还讲述了创建单元测试的过程，并讨论了针对纯粹的 TDD 方法与该方法的实际运用之间的折中。

在该章最后，我们将焦点放在依赖昂贵资源的单元测试模块上。基于一个使用了流行 mocking/stubbing 架构的示例，引入了模拟对象和存根对象的概念。

**第 5 章：进一步简化——控制反转**

第 5 章回顾了第 3 章引入的概念，集中介绍了有助于使企业开发更加容易的一些工具和模式。在讨论了工厂(Factory)和服务定位器(Service Locator)的优缺点之后，介绍了控制反转准则和用于描述设计的抽象准则，相比于过程式程序设计，在这些设计中系统的流程是倒置的。该章最后介绍了一个自建的简单控制倒置容器，并简要回顾了一个称为结构映射(Structure Map)的流行开源产品。

**第 6 章：进入关注中心**

在结束对与企业开发相关的核心概念和编码基本原理的讨论后，现在，读者要进入本书的模式部分。在此，我们讨论了一些流行的设计模式，以及一些可用于构建良好设计的企业系统的架构。这是本书中第一个涉及中间件概念的章节。首先回顾了分层设计的历史，从原有的主机系统模型开始，逐步进入客户-服务器架构和 Web 开发。然后，探讨了一些流行的设计模式，这些模式现在应用于分布式系统中，例如面向服务和面向消息的中间件。

**第 7 章：编写自己的中间件**

第 7 章集中讨论了包含在应用程序中的业务逻辑。

我们讨论了 3 种流行的中间件模式：事务脚本(Transaction Script)、活动记录(Active Record)和领域模型(Domain Model)模式。随后，简要地探讨了领域模型设计(Domain Model Design)，这是当前获得相当流行度的一种设计方法。这种新兴模式主要关注的是应用程序的业务逻辑，它强有力地分离了对技术基础设施的关注。

在第 7 章的结束处，构建了一个针对虚构的按揭贷款批准(Mortgage Loan Approval)应用程序的领域模型，在后续各章中还会再次用到这个项目，其中用到了领域驱动设计的一些核心概念，对于该项目，需要经历几个过程，包括收集需求、与领域专家对话，以及基于测试驱动开发的原理构建领域模型。

**第 8 章："挖掘"自己的业务**

第 8 章讨论了持久化(persistence)的概念。该章首先简要地讨论了数据访问层(data access layer)的职责，以及与 ADO.NET 集成的传统方法。然后介绍了对象关系映射器(Object Relation Mapper)，接着讨论了利用架构而不自己实现架构的优缺点。

然后，该章介绍了两种持久管理的方法。第一种方法是一种数据模型方法，它使用了 Microsoft 的 LinqToSQL。第二种方法是一种模式，它使用了 Microsoft 的实体架构对象关系映射器，该映射器具有一种模式，称为事件访问对象模式。

在第 8 章的结束处，提供了第 7 章创建的抵押贷款核准应用程序，并用 nHibernate 构建了一个数据库层。

**第 9 章：组织前端**

继续我们对系统设计的探讨，该章探讨用户界面开发领域。与第 6 章一样，该章首先讨论用户界面设计和编程的历史。该章向读者介绍了快速开发设计的一些影响和一些可用于支持快速开发设计的平台工具。经过拖放屏幕设计的阶段并进入 Web 程序设计阶段，我们讨论了一些较为流行的新兴设计模式，它们最终成为了企业设计的组成部分，例如模型-视图-控制器(Model-View-Controller)模式。

**第 10 章：模型-视图-表示器模式**

第 10 章直接进入用户界面(User Interface，UI)设计，深入回顾了模型-视图-控制器模式。创建可测用户界面要求运用松散耦合的编码模式。然而，如果不对应用程序进行分解，可能很难将该模式应用于早期的用户界面代码。正确的模式应该能够对用户界面代码进行单元测试，而无须过度干扰原来的代码。这正是模式-视图-表示器(Model-View-Presenter，MVP)模式的目标。事实证明，这可能是用于企业用户界面设计最流行的方法之一，模型-视图-表示器模式有助于将界面代码与可视实体本身分离。第 10 章详细地介绍了这种模式，并逐步介绍了各个组件并解释它们的意图，回顾了这些组件提供的关于可测性的各个方面。该章提供了一段详细的代码示例，用于说明用模型-视图-表示器模式构建的一个简单的按揭贷款计算器，同时给出了针对 Web 和胖客户平台的情况作为示例。

**第 11 章：模型-视图-控制器模式**

沿着第 9 章和第 10 章列出的主线，该章深入讲解了模型-视图-控制器模式。对该模式的历史进行了简要回顾，并着重介绍了 Ruby on Rails 架构(简称 RoR 架构，David Heinemeier Hansson 编写的一个流行架构)，以及它所具有的影响。向读者介绍了 ASP.NET MVC，即 Web 模型-视图-表示器模式，它是由 Microsoft 最近发布的，用于在 Windows 平台上开发 Web 应用程序。接着详细讨论了该模型展现的核心组件和行为。回顾了 Web MVC 模型的优缺点，并将其与第 10 章中介绍的模型-视图-表示器模式进行了特别比较。然后，该章内容就转入一段具有挑战性的代码示例，该代码使用了第 7 章和第 8 章中介绍的按揭贷款模型，在第 10 章中也引用了该模型。

**第 12 章：组合所有内容**

作为本书的总结，综合性地概括了前面各章中讨论过的所有概念、方法和设计模式。并回到第 1 章中介绍的核心价值，对前面每一章进行了简要回顾，总结了每一章所传达的关键点，并将这些关键点与更广泛的企业目标关联起来。讨论了如何将恰当的模式用于恰当类型的系统，仅选择适合系统需求的架构和模型。该章最后简要地浏览了一个大概的、多面的抵押应用程序，该应用程序使用了本书探讨过的大多数主题和模式。

## 本书的组织

本书旨在作为一个逐步深入的向导和一个连贯的参考资源。本书可以分为不同的部分，既可以将每个部分作为独立的整体来阅读，也可以作为片段来阅读。本书的第 I 部分

讨论了企业开发背后的基本原理。第II部分对编码模式进行了更深入的讨论。主要涉及松散耦合的概念、对现有代码解码的最佳途径和测试驱动设计的优点。第III部分(也是最后一个部分)对企业系统中使用的一些较为通用的设计模式进行了简要回顾。回顾了结构化中间件的流行方式，介绍了数据挖掘和持久技术，并提供了关于企业 UI 设计的一些背景。每章都是从解释主题开始，并且大多数情况下都带有详细的代码示例。代码示例可以单独浏览，或者为了更全面地进行系统评估，也可以将它们组合在一起进行研究。

## 源代码

读者在阅读本书提供的代码时既可以亲自键入所有代码，也可以使用随书提供的代码文件。本书所有代码均可以从 http://www.wrox.com 网站下载。进入该网站后，请读者根据本书的书名查找本书(读者既可以使用搜索框进行查找，也可以使用书名列表进行查找)，然后单击本书详细内容页面上提供的 Download Code 链接，就可以下载本书提供的所有代码。

注意:

因为许多书籍名称与本书类似，因此读者也可以通过 ISBN 进行查找，本书的 ISBN 为：978-0-470-44761-1。

下载代码后，读者可以利用一种压缩工具将代码解压。此外，读者还可以通过访问网站 http://www.wrox.com/dynamic/books/download.aspx 中提供的 Wrox 代码下载页面来获取本书提供的代码，也可以下载 Wrox 出版的其他书籍提供的代码。

## 勘误表

为了避免本书文字和代码中存在错误，我们已经竭尽全力。然而，就如世界上不存在完美无缺的事物，本书仍然可能存在错误。如果读者在我们编写的书籍中发现了诸如拼写错误或代码缺陷等问题，那么请告诉我们，我们对此表示感谢。利用勘误表反馈错误信息，可以为其他读者节省大量时间，同时，我们也能够受益于读者的帮助，这样有助于我们编写出质量更高的专业著作。

如果读者需要参考本书的勘误表，请在网站 http://www.wrox.com 中用搜索框或书名列表查找本书书名。然后，在本书的详细内容页面上，单击 Book Errata 链接。在随后显示的页面中，读者可以看到与本书相关的所有勘误信息，这些信息是由读者提交、并由 Wrox 的编辑们加上的。通过访问 www.wrox.com/misc-pages/booklist.shtml，读者还可以看到 Wrox 出版的所有书籍的勘误表。

如果读者没有在 Book Errata 页面上找到其发现的错误，那么请读者转到页面 www.wrox.com/contact/techsupport.shtml，针对您所发现的每一项错误填写表格，并将表格发给我们，我们将对表格内容进行认真审查，如果确实是我们书中的错误，那么我们将在该书的 Book Errata 页面上标明该错误信息，并在该书的后续版本中改正相关错误。

# 关于 p2p.wrox.com

如果读者希望能够与作者进行讨论，或希望能够参与读者的共同讨论，那么请加入 p2p.wrox.com 的论坛。这个论坛是一个基于 Web 的系统，读者可以在论坛发表与 Wrox 出版的书籍有关的技术信息，并与其他读者和技术用户进行讨论。论坛提供了订阅功能，可以将与读者所选主题相关的新帖子定期发送到读者的电子邮箱。Wrox 的作者、编辑、业界专家以及其他读者都会参与论坛中的讨论。

读者可以在 http://p2p.wrox.com 参与多个论坛的讨论，这些论坛不仅能够帮助读者更好地理解本书，还有助于读者更好地开发应用程序。如果读者希望加入论坛，那么请读者按照以下步骤执行：

(1) 进入 http://p2p.wrox.com 页面，单击 Register 链接。

(2) 阅读使用条款，然后单击 Agree。

(3) 填写必要的信息(必要时也需要填写可选信息)，然后单击 Submit。

(4) 随后读者会收到一封电子邮件，邮件中说明了如何验证帐号并完成整个加入过程。

**注意：**

*要阅读论坛信息，读者无须加入 P2P。但是如果读者需要发表主题或发表回复，那么读者必须加入论坛。*

成功加入论坛后，读者就可以发表新主题了。此外，读者还可以回复其他主题。读者在任何时间都可以阅读论坛信息。如果读者需要论坛将新的信息发送到自己的电子邮箱，那么可以单击论坛列表中论坛名称旁的 Subscribe to this Forum 图标完成该功能设置。

如果读者需要获得更多与 Wrox P2P 相关的信息，请阅读 P2P FAQs，这样可以获得大量与 P2P 和 Wrox 出版的书籍相关的具体信息。阅读 FAQs 时，请单击 P2P 页面上的 FAQs 链接。

# 设定目标

如果您是企业.NET 编程领域的开源领军人物，勤于写博客和在论坛中冲浪，精于企业开发，那么您可以立即放下本书。本书并不是一本涵盖关于企业的所有内容的书籍。本书不要求完美的、测试优先(Test First)的方法。本书并不处处吹捧敏捷至上(Agile Manifesto)，而且它对企业完美主义者也没有吸引力。

这是一本旨在向读者介绍良好系统设计的优点以及如何将其应用于大型系统项目的书。然而，我们当然不愿意招致企业界和开源界的愤怒，本书的目标是程序设计领域的其他方面，注意到这一点很重要。我们的意图在于提供一个关于现代软件设计原理的更简单、更灵活的开端。尽管本书是专门针对 Microsoft 程序员而设计的，但是，在此讨论的实践和方法适用于任意技术环境中的所有软件开发，理解这一点很重要。在此使用的很多工具和架构是由其他开发平台演变而来的，比如 Java 和 Ruby on Rails。其中还有一些工具来自于

Microsoft，但是在此提到它们不仅仅是为了支持基于 Microsoft 平台的软件开发，而完全是为了设计良好的软件。本书的两位作者都是开源领域的积极支持者。二人都坚定地信奉：一个问题的最佳解决方案并不一定总是来源于同一个公司。

在接下来的几个月中，您一定会发现大量具有类似主题的书籍，其中很多都是专门针对 Microsoft 解决方案的。本书首先是一本关于理解良好系统设计的书，其次才是关于选择最适合您需求的途径的书。我们会讨论 Microsoft 技术的一些长处，同时还会讨论一些显著的弱点。理解各种不同的可用工具的优缺点有助于使您针对自己的项目做出更好的决策，并最终获得更佳的系统设计。

# 目　录

# 第 I 部分

# 实用企业开发介绍

第 1 章：企业设计概念

第 2 章：企业代码