PROGRAMMER TO PROGRAMMER™

.NET Enterprise Development in C#:
From Design to Deployment
Written and tested for final release of .NET v1.0

.NET 企业应用高级编程

——C# 编程篇

Matthew Reynolds Karli Watson 著 康博 译

清华大学出版社
http://www.tup.tsinghua.edu.cn

.NET 企业应用高级编程

——C#编程篇

Matthew Reynolds
Karli Watson 著

康 博 译

清华大学出版社

(京) 新登字 158 号
北京市版权局著作权合同登记号：01-2002-0257

内 容 简 介

C#是 Microsoft 专用于.NET Framework 平台的一门新型编程语言。虽然很多语言都能够编写.NET 代码，但 C#是惟一针对.NET Framework 而设计的语言，因此在今后的几年内，C#将会成为编写.NET 应用程序的首选。本书以通用企业软件的开发示例为基础，逐步阐明了开发企业应用程序的方法和技巧。首先介绍了开发企业软件的基础知识，然后介绍了如何通过 Web 服务发布数据以及为移动设备开发应用软件，还讨论了如何加强.NET 企业应用程序的安全性问题，最后为读者介绍了管理、监控、调试和优化企业应用程序的方法和技巧。

本书适用于从事企业应用程序开发的中高级程序员，也有助于其他读者学习如何开发企业应用程序的相关知识。

Matthew Reynolds，Karli Watson：.NET Enterprise Development in C#: From Design to Deployment
EISBN：1-86100591-1

书　　名：.NET 企业应用高级编程——C#编程篇
作　　者：Matthew Reynolds，Karli Watson 著　　康博 译
出 版 者：清华大学出版社 (北京清华大学学研大厦，邮编：100084)
　　http://www.tup.tsinghua.edu.cn
责任编辑：李 阳
封面设计：康 博
版面设计：康 博
印 刷 者：北京密云胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092　1/16　印张：25.5　字数：652 千字
版　　次：2002 年 8 月第 1 版　2002 年 8 月第 1 次印刷
书　　号：ISBN 7-302-05728-1/TP•3380
印　　数：0001～5000
定　　价：48.00 元

出版者的话

近年来，国内计算机类图书出版业得到了空前的发展，面向初级用户的应用类软件图书铺天盖地，但是真正有深度和内涵的高端图书不多。已经掌握计算机和网络基础知识的人们，尤其是 IT 专业人士迫切需要“阳春白雪”。IT 图书市场呼唤精品！

为了满足这种市场需求，清华大学出版社从世界出版业知名品牌 Wrox 出版公司引进了受到无数 IT 专业人士青睐，被奉为 IT 出版界经典之作的 Professional 系列丛书。这套讲述最新编程技术与开发环境的高级编程丛书，从头到尾都贯穿了 Wrox 出版公司“由程序员为程序员而著(Programmer to Programmer)”的出版理念，每一本书无不是出自软件大师之手。实际上，Wrox 公司的图书作者都是世界顶级 IT 公司(如 Microsoft，IBM，Oracle 以及 HP 等)的资深程序员，他们的作品既深入研究编程机理，传授最新编程技术，又站在程序员的角度，指导程序员拓展编程思路，学习实用开发技巧，从而风靡世界各地，被 IT 专业人士和程序员视为职业生涯中的必读之作。

为了保证该系列丛书的质量，清华大学出版社迅速组织了一批位于 IT 开发领域前沿的专家学者进行翻译，经过编辑人员的进一步加工整理后，现陆续奉献给广大读者。

读者可以从 www.wrox.com 网站下载所需的源代码并获得相关的技术支持。同时，也欢迎广大读者参与 p2p.wrox.com 网站上的在线讨论，与世界各地的编程人员交流读书感受和编程体验。

序　言

.NET Framework 是 Microsoft 提供的一个新型的编程环境。.NET Framework 的设计目标是：通过改变服务、设备和用户交互的方式使 Internet 在商务中的应用得到革新。使用.NET Framework 构建的下一代应用程序将允许用户同计算机之间以及计算机同计算机之间的交互更加方便、更加广泛、更加全面。

.NET 除了提供了与设备交互的新方法之外，还为我们提供了创建这些设备的快捷而又有效的方法。通过使用像 XML 这样的业界标准，开发人员能够很容易地构建出跨设备和领域边界的应用程序。当然，企业必须保证其信息的安全，否则.NET Framework 所提供的这些技术还不是非常完善。Microsoft 在开发.NET 技术的过程中一直注意了这一点，并提供在 Internet 上保存数据、允许应用程序为用户提供定制功能的安全方法。

本书论述的重点是设计和开发完整企业应用程序的解决方案。通过阅读本书中一个扩充的案例分析，读者可以对在设计和开发完整企业应用程序中所遇到的问题有一个清晰的认识。并对如何创建.NET 企业应用程序，以及如何通过各种设备、使用不同方法并以一种安全的方式与企业应用程序相连接等问题有一个广泛且基本的了解。因此，无论需要创建什么规模的应用程序，阅读本书将会帮助您了解如何着手去做，以及如何更清楚地定义企业应用程序各个部分的广度和深度。

本书的主要内容

第 1 章介绍了本书所涉及的主要内容，并对贯穿本书的应用程序进行了概述。为了给本书后面的示例开发提供铺垫，还以常规的方式对分布式编程和 N 层应用程序进行了特别的介绍。

第 2 章讨论了应用程序的设计。我们将对 Wrox 企业对象(Wrox Enterprise Objects，WEO)这个通过 Builder 工具(见附录 A)创建的丰富的对象层进行全面的介绍。接下来，还将介绍如何使用这个工具，以及如何创建和使用存储过程。

在了解了如何创建和运行基本的应用程序之后，第 3 章将带领您学习创建一个应用程序浏览器。首先简要地介绍使浏览器可用于一般应用程序的各种方法，然后展示如何创建浏览器，这里主要涉及到了诸如身份验证、添加菜单选项以及运行子应用程序等内容。

第 4 章将介绍如何创建一个桌面应用程序，以及如何将该桌面应用程序与业务对象挂钩。本章通过一个相对简单的示例论述了桌面应用程序的原理。通过该示例，为您展示了如何编辑客户信息，以及如何使用应用程序在数据库中查找客户。

第 5 章讨论了自动化处理。主要介绍了捕获、传送、处理以及报告指令，并展示如何创建指令处理器。在讨论这个主题时，我们还会介绍一下有关体系结构的内容，以及轻松地开发这组功能的方式。

第 6 章对 Web 服务进行了讨论。该章内容展示了如何使用 Web 服务来提供业务对象的功能。首先，我们介绍了通过桌面应用程序来使用 Web 服务，在第 7 章中，将介绍如何通过 ASP.NET 来使用 Web 服务。还将通过一种安全的方式来给出一个示例，即使用 SSL 来为 IIS 中的身份验证服务传送用户信息的方式。

由于目前可以查看来自不同客户的数据，包括无线客户，因此第 8 章将介绍如何使用移动客户连接应用程序。我们将特别介绍 Microsoft 的 Mobile Internet Toolkit，然后示范一个使用了企业业务对象的小型应用程序。

第 9 章讨论了有关服务的内容。首先介绍用于验证远程对象的用户身份的基于令牌的身份验证系统。本章将介绍有关创建一个服务、与该服务连接方面的内容，最后还介绍了如何安装并运行这个服务。

第 10 章讨论了安全性和自动化部署。首先，深入讨论了代码访问安全性方面的问题，其中包括了获得凭证及评估许可等内容。然后讨论应用程序的安全性和保护客户编辑。

第 11 章深入介绍了有关管理的内容。在这一章，我们构建了一个管理工具，并且展示了一些同样适用于其他情况的基本原理，例如停止和开始服务。最后，简要介绍了一下 Microsoft 的管理控制台(Microsoft Management Console)。

本书的最后一章，第 12 章介绍了管理企业应用程序方面的内容，主要讨论的主题包括执行计算、异常报告、调试和负载平衡。为了使本书的内容更加完整，我们还将简要介绍一下 Microsoft Application Center 2000。

附录 A 介绍了书中一直使用的 Builder 工具的内部工作情况，以帮助读者对后台的运行情况有一个更好的认识。

本书读者对象

本书适合于那些对开发和出售通用软件产品感兴趣的读者，这些软件产品可以用作业务线应用程序(或在某种程度上用作生产率应用程序)。那些为自己公司定制通用产品，或者是为其他公司定制通用产品，并将其作为业务线应用程序来部署的人们也可以从本书中获取大量有价值的信息。

当然，所有希望深入了解企业应用程序，以便更好地探究其内部运行情况的人们也会从本书中受益。令人欣喜的是，本书作者开发了一个 Object Builder 工具，该工具允许我们基于数据创建对象。

本书是一本中级水平的程序开发类图书，可以帮助初学者和高级开发人员学习创建企业应用程序的相关知识。

学习本书所需要的准备工作

为了能够运行本书中的示例，必须具备以下环境：

- Windows 2000 或 Windows XP

- Visual Studio .NET

用户支持

我们一贯重视听取读者的意见，特别希望知道您对本书的反馈，例如，您喜欢和不喜欢的内容，以及您认为我们下次如何才能做得更好等方面的信息。您可以将意见寄给我们，或者发电子邮件到 feedback@wrox.com。请您在反馈中务必写清楚本书的书名。

如何下载本书的示例代码

在您访问 Wrox 公司站点(地址为 http://www.wrox.com/)时，通过 Search 工具或书名列表，可以方便地定位您所需要的书目。然后，单击 Code 栏中的 Download 超链接，或者单击本书的详细页面中的 Download Code 超链接，就可以下载相应的示例代码。

从站点中下载的文件都是使用 WinZip 压缩过的文档。将附件保存到本地磁盘上的文件夹中后，需要使用一个解压缩程序(例如 WinZip 或 PKUnzip)来解压缩文件。在解压缩文件时，通常将代码解压缩到每一章所在的文件夹中。在解压缩的过程中，应确保解压缩程序(WinZip、PKUnzip 等)被设置为使用原有文件夹名。

勘误表

我们已经尽最大努力确保本书中的文本和代码没有错误，但是错误肯定还是在所难免。如果您发现本书存在错误，例如拼写错误或不正确的代码段，请给我们发来反馈信息，我们将不胜感激。勘误表的发送可以节约其他读者学习本书的时间，而且能够帮助我们提供更高质量的信息。请将您的反馈信息用 E-mail 发送到 support@wrox.com。我们将检查您的反馈信息，如果正确，将被粘贴到本书的勘误页面上，或者在本书的后续版本中使用。

要在站点上找到勘误表，请访问 http://www.wrox.com/，并通过 Advanced Search 或者书名列表轻松定位本书页面。然后，单击 Book Errata 超链接即可。

E-mail 支持

如果希望直接向了解本书的专家咨询书中的问题，可以发送电子邮件到 support@wrox.com，要求在邮件的主题中带上本书的书名和 ISBN(国际标准图书编号)的后 4 位数字。一封典型的电子邮件应包括以下内容：

- 在主题中必须有本书的书名、ISBN 的后 4 位数字和问题出现的页码。
- 信息的主题应包括读者的名字、联系信息和问题。

我们将不发送给您无用的邮件，因为，我们仅仅需要有用的详细资料，以便可以节约您和我们的时间。当您发送一个电子邮件信息时，它将得到下面一系列的支持：

- 用户支持。首先，您的信息将被递送到用户支持人员手中，由他们先进行阅读。归档一些被频繁问到的问题，并立即回答有关本书或者 Web 站点的任何常见问题。
- 编辑支持。接着，一些有深度的问题将被送到对本书负责的技术编辑手中，他们在程序设计语言或者特定的产品上有着丰富的经验，能够回答相关主题的详细技术问题。

● 作者支持。最后，如果编辑不能回答您的问题(这种情况很少发生)，则他们将请求本书的作者。我们将尽量保护作者免受干扰，以便不影响其写作。然而，我们也非常高兴转寄给他们一些特殊的问题。所有 Wrox 公司的作者都为他们的书提供技术支持。作为回应，他们将发送电子邮件给用户和编辑，从而使所有的读者受益。

Wrox 公司的支持过程仅仅对那些与我们出版的书目内容直接相关的问题提供支持，对于超出常用书目支持的问题，您可以从 http://p2p.wrox.com/ forum 中的公共列表中获得支持信息。

p2p.wrox.com

为了便于作者和其他人讨论，特将讨论内容加入到 P2P 站点的邮件列表中，而且我们惟一的系统将 programmer to programmer™(由程序员为程序员而作)的编程理念与邮件列表、论坛、新闻组以及所有其他服务内容(一对一的邮件支持系统除外)相联系。如果您向 P2P 发送一个问题，它一定会被登录邮件列表的 Wrox 公司作者和其他相关专家检查到。无论您是在阅读本书，还是在开发自己的应用程序，都可以在 p2p.wrox.com 站点中找到许多对自己有所帮助的邮件列表。

按照下面的步骤可以预定一个邮件列表：

(1) 登录 http://p2p.wrox.com/站点。

(2) 从左边的主菜单栏选择一个适当的类别。

(3) 单击希望加入的邮件列表。

(4) 按照说明订阅并填写自己的邮件地址和密码。

(5) 回复您收到的确认邮件。

(6) 使用预定管理程序加入更多的邮件列表并设置自己的邮件选项。

本系统提供最好支持的原因

您可以选择连接到整个邮件列表，也可以只接收每周的邮件摘要。如果您没有时间和工具来接收邮件列表，可以直接查找我们的在线文档。独特的 Lyris 系统可以将一些没用的垃圾邮件删除，并保护您的电子邮件地址不被侵扰。当存在加入和离开列表、以及任何有关列表的其他常见问题时，请发送邮件到 listsupport@p2p.wrox.com。

目　录

第1章 引 言

在本书中，要建立一个完整的企业应用程序，该应用程序贯穿于本书始终，该程序或许会被某个书商使用。正如我们所做的，我们希望通过创建这样的一个应用程序来论证它能够达到怎样的一个业务领域，以及如何解决诸如数据并发、版本化和安全性等这样的基本问题。本书还将介绍已开发的 Object Builder 系列工具，这类工具有助于软件应用程序的开发。通过本书所介绍的技术，我们使用了一个基本的示例来说明创建企业软件应用程序的方法。学习本书，有助于您将本书所介绍的技术应用于自己的应用程序中，并且能够帮助您针对自己的业务设计和构建解决方案。如果想马上简单地获取本书中的代码样例并使用它们是不现实的。不过，这些代码是您构建自己应用程序的基础。

最近，我们发现这样一种趋势，即孤立的应用程序已经向更为分布式的模型发展了，这种更为分布式的模型使用的是 Internet 技术——尤其是 Web 浏览器和 Web 服务器。然而，由于某些因素(将在后面讲到)的影响，在今后几年中，桌面应用程序很可能会再次有大的发展。因此，本书将重点介绍桌面应用程序的开发以及基于内部网和外部网(intranet/extranet)的解决方案。当然，还要介绍非桌面应用程序，尤其是 Web 以及 Web 服务应用程序。

本书要介绍的构建企业应用程序的主要内容包括：

- 含有丰富功能的可用业务对象的 N 层体系结构
- 桌面(Windows Forms)开发
- Web 开发(包括 intranet/extranet 和 Internet)
- 访问移动设备
- Web 服务
- 部署
- 集中控制和管理
- 安全性

在介绍应用程序之前，我们先简要地说明一下企业应用程序的背景。

1.1 变化中的企业面貌

很多现代公司发现，他们所处的商业环境在过去的几十年中发生了根本性的变化。随着本国及全球市场的开放，竞争变得日趋激烈。为了生存，企业必须在业务处理中利用更多、更好的工具。最近的技术进步为企业带来了更高的效率，并且为客户得到他们想要的产品和服务提供了廉价且方便的新途径。

工作领域应用 IT 技术的早期，应用程序被局限在单个的计算机上，这极大地限制了计算机解决问题的领域。很多早期的桌面解决方案需要存储在每台机器上的公司数据库的本地副本的支

持，在这些机器上，用户可以独立地访问或操作它们。这个过于简单化的模型带来了大量的维护方面的问题。例如， 必须在某个合适的时间(通常是在最后的一个营业日)整理每一个本地副本的变化，然后，必须将更新后的数据库重新发布到各个工作站。随着技术的进步，应用程序变得更为复杂了，逐渐出现了这种组装的解决方案，该解决方案基于在整个网络中都可以访问的在线的中央数据库。执行用户请求任务的工作负荷是由台式计算机和服务器共同分担的：这种模型被称为分布式运算模型。

1.2 分布式应用程序

很多现代企业的解决方案都包含了若干个分布式应用程序，在这些应用程序中，工作负荷是根据任务的类型以及客户机的处理能力由客户机和服务器共同分担的。

那些运行在高速局域网中的应用程序迫切需要被扩充，以便获得通往广阔 Internet 的入口。由于不存在部署问题，所以，要调用的那些诸如 intranet/extranet 的应用程序是非常吸引人的。要使基于 intranet 的应用程序基本可用，您只需为 Web 浏览器提供一个合适的用户接口，以及一种连接到服务器上的方法即可(对于不同 Web 浏览器的性能，在这里便不深入介绍了)。简单而轻松的部署使得 internet/extranet 应用程序变得非常吸引人，这同时也解释了最近所看到的，应用程序为什么朝着这种企业应用程序的方向发展的原因。

这种应用程序也存在着不足，为了迎合 Web 浏览器的性能及连接服务器的限制，需要在它们之间作出折衷，这样就可能限制了应用程序的功能和可用性。另一点需要考虑的是，这种方式极大地忽略了现代台式计算机的整体性能，虽然现代台式计算机的处理能力得到了极大的提升，但由于所有应用程序的处理实际上都发生在服务器上，所以，计算机本身强大的处理能力并没有利用到。

上述的这种情况导致了众所周知的胖客户(rich client)的发展。在胖客户机上，客户应用程序变得更加复杂，这主要取决于客户机自身的性能。这种(richclient)机能够独立地执行多个任务。

- 瘦客户——这种客户机仅仅显示服务器发出的数据、获取用户的输入信息，然后将其发送回服务器。
- 胖客户——这种客户机能够做很多处理数据方面的工作。例如，服务器可能会返回一系列原始数值，然后，客户机可以将这些原始数值转变为一个图形。胖客户能够在将用户输入的数据发送到服务器之前先对其进行验证等。

实际上，对瘦客户和胖客户我们很难明显地进行划分，因为，存在了许多划分两者的标准，不过，基本的划分原则仍然会保留的。也许，将客户机划分为胖客户最有力的论据是，它在本地执行了大量共享的处理，从而使需要服务器处理的工作减少了，因此，也就允许服务器同时支持更多的用户。

客户机/服务器这个术语对于目前业界中的网络配置来说过于简单化了，但是，区分客户机与服务器的基本概念仍然有效。很多这样的应用程序仍然可以看作是客户机/服务器应用程序，尽管基本的模型(比如我们即将介绍的 n 层模型)变得更加复杂。

1.2.1 N层模型

Microsoft并没有创造n层概念，不过，n层概念早已成为Microsoft软件开发的一大部分。因而，很多人都认为它是属于Microsoft的，并且是由Microsoft推广的。实际上，n层只是一种将应用程序划分为几个部分(也称为层(tiers或layers))的简单方式，每个部分执行特有的工作。

N层是客户机/服务器技术的延伸。在这种模式中，一台服务器向多台客户机提供数据。(虽然这里说是“一台服务器”，但可能不止一台，很多情况下一个应用程序会使用到多台服务器。尽管把客户机/服务器技术当作“一台服务器，多台客户机”来理解有点欠妥，但还是可以帮助我们理解这一概念)。

作为在大型机环境中开发、运行软件这种方式的自然延伸，客户机/服务器技术最初便是由此发展而来的。所谓的大型机环境是指，配置了功能强大的中央处理器的一些大型机负责执行应用程序所请求的处理行为，而处理能力小且较小的一些计算机负责与用户的交互。随着小型计算机(具有更低的价格、更强大的功能)的不断增加，大型机逐渐失去对用户的吸引力，用户越来越趋向于使用小型机了。伴随而来的结果是，用户端(或称客户端)计算机变得越来越便宜、且功能更强大，这就意味着中央计算机不得不做更少的工作了。

随之发生的是客户端代码变“胖”了，从理论上来说，服务器端的代码应该是变得更“瘦”了。然而，随着客户端软件变得更加复杂，部署客户端软件的工作也就相应出现了更多的问题。围绕这一点，Internet技术(尤其是Web服务器和Web客户机)显得更为完善，这意味着我们又转回到了与大型机途径非常相似的一种环境，即一台功能强大的中央服务器和多台不太复杂的客户机(也可称为哑终端)。

目前，在使用.NET 的情况下，我们可以看到软件开发和运行的方式又重新向着“稠化(thickening)”的客户端代码转变了，这意味着将有更多的处理过程发生在客户机上。为了使软件的部署变得更加容易，Microsoft针对不同的技术改进做了许多工作，因此，这种转变是极有可能发生的。在第10章中将介绍与此相关的内容。

今天，大多数的分布式应用程序都建立在3层结构的基础上，即数据层、业务层和表示层(Data、Business、Presentation)。如果从客户机/服务器的观点来看，数据层和业务层可以被看作是服务器端的，表示层则可以看作是客户端的(尽管在Web开发中表示层的任务是由服务器和客户机共同分担的，但是我们仍然将表示层看作是客户端的)。

- 数据层——该层负责从数据库中提取数据并将其提供给业务层，或者从业务层中提取数据并将其提供给数据库。
- 业务层——该层负责应用程序的实际处理。例如，它负责确保新的订单已正确地存储在数据库中，以便用户能够查找客户信息等。
- 表示层——该层负责在业务层和用户之间编组数据。

传统的3层设计从以下两点来看是有用的：它使最初的应用程序的创建变得更为容易，并且使应用程序的扩充也变得更加容易。下面便分别介绍一下数据层、业务层和表示层。

1. 数据层

如果依次考虑每一层的话，数据层常常正好是SQL服务器，或是根据您的应用程序所选择的DBMS。在企业环境中，您经常会发现企业已经为您做出了DBMS方案的选择。事实上，在

企业环境中，您会发现您被指派使用 Oracle 进行应用程序的开发(Oracle 是世界上最大的数据库软件开发公司，而 Microsoft 是最大的软件开发公司)。在本书中，使用的数据库系统是 Microsoft SQL Server 2000。

当设计应用程序时，应确定需要存储什么数据以及存储数据的最佳方式。这就会涉及到构建数据库模式以及向数据库中加载数据(假定您有一些构建应用程序时需要使用的数据，或者至少有一组测试数据)。设计数据层的任务通常是由合格的数据库管理员和以技术工程师身份工作的开发人员共同承担的。

应用程序中数据层的实际定义其实有点含糊不清。虽然 SQL Server 本身明确地构造了数据层，但是，用于和 SQL Server 通信的各种对象存在于数据层部分和业务层中。例如，与 SQL Server 进行物理通信的 ADO.NET 类可以看作是“数据层”代码，尽管它们与业务层代码一样运行在相同的进程中。通常，派生大多数业务层类的类(这些类我们称之为 Enterprise Objects)可以被认为是数据层的类或是业务层的类，这主要取决于您的观点，尽管它们也都在业务层代码中运行。

2. 业务层

软件设计过程真正变得有趣是从业务层开始的。您在业务层所做的工作就是构建软件，使之与来自管理小组不同部分的形式化了的过程，以及遵循业务本身的进程相匹配。构建业务层是模块化这些业务进程结合的产物，同时也要关注面向对象技术和组件(面向对象的技术和组件是构建一个丰富的对象模型的基础，开发人员可以在自己的应用程序中借助它们)。

对业务进程稍加考虑，我们就会发现：

- 货物一经收到，就要对其进行核对并装入仓库的相应位置。
- 当客户返回一个商品项时，他们可以凭最初的订单或信用证在 30 天内获得全额退款。
- 当一个职员要预定休假时间，则该部门的其他任何职员不得同时在此期间休假。

总之，所有的业务都要被模型化为进程或过程。如果要使进程自动化或计算机化，应用程序的业务层就必须严格遵循这个过程。

这种模型如此流行的主要原因是(假设我们现在讨论的是一家管理良好的公司)业务过程很少有变动。正规的公司不会每隔一天就修改自己的退货政策。只有在其内部各个经理及决策者磋商后才会改变其政策。

假如有一个管理客户退货的应用程序，该应用程序使用了业务层中的对象来处理退货。因为这些业务层中的对象必须遵循公司的过程，所以，这就有效地保证了使用应用程序的用户也必须要按照公司的程序办事，任何“非法”的行为都会被禁止。另外，如果公司的程序有所变化，那么就要按照新规则来修改业务层的代码。一旦修改后的代码生效，该应用程序就会“感受”到新规则的效果，使用该程序的用户就会被禁止做任何与新规则不符的事情。

在本书介绍的这个应用程序中，我们将对一些业务过程进行模型化。很多过程都是比较基础的—— 鉴于本书是一本中级难度的软件开发类书籍，我们尽量对所介绍内容的复杂程度进行了限制，不过，我们仍会介绍其中的逻辑步骤。

3. 表示层

另一个推动 n 层结构应用的重要因素是，表示层可以由多个应用程序组成，而且，每个应用程序分别是为不同的平台和目的定制的，同时，用户仍会从同样的业务层代码库中受益。例