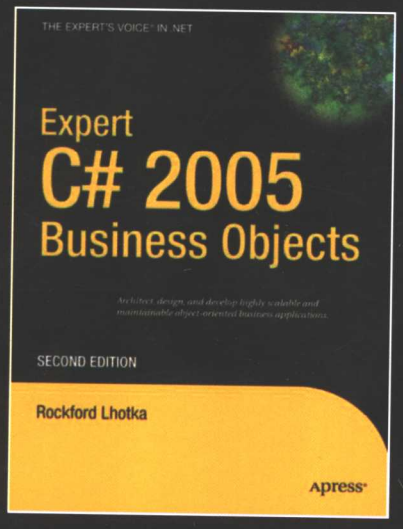
Broadview
WWW.BROADVIEW.COM.CN

THE EXPERT'S VOICE® IN .NET

Apress®

Expert C# 2005 Business Objects

Expert C# 2005 Business Objects 中文版（第2版）



架构、设计和开发极具可扩展性和可维护性的面向对象商业应用

[美] Rockford Lhotka 著
王鑫 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn


Expert
C# 2005
Business Objects

Expert C# 2005 Business Objects 中文版（第2版）

Expert C# 2005 Business Objects Second Edition

[美] Rockford Lhotka 著

王 鑫 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书描述了怎样应用面向对象的概念来进行.NET 应用程序的架构、设计和开发。作者将重点放在了面向业务的对象，即业务对象和怎样在包括 Web 和客户机/服务器结构的不同分布式环境中来实现它们。本书使用了大量的.NET 技术，面向对象的设计与编程思想，以及分布式架构。本书的前半部分叙述了如何在.NET 环境创建这个框架来支持面向对象的应用程序开发的流程，后半部分应用这个框架创建了一个带有几个不同接口的示例应用程序，本书适合 C#应用开发人员阅读。

1-59059-632-3 Expert C# Business Objects by Rockford Lhotka

版权贸易合同登记号　图字：01-2005-2688

图书在版编目（CIP）数据

Expert C# 2005 Business Objects 中文版 /（美）霍特卡（Lhotka,R.）著；王鑫译.—北京：电子工业出版社，2007.6
书名原文:Expert C# Business Objects
ISBN 978-7-121-03818-1

I. E… II.①霍…②王… III.C 语言—程序设计 IV.TP312

中国版本图书馆 CIP 数据核字（2007）第 010842 号

责任编辑：梁　晶
印　　刷：北京东光印刷厂
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
经　　销：各地新华书店
开　　本：787×1092　1/16　　印张：43.75　字数：840 千字
印　　次：2007 年 6 月第 1 次印刷
定　　价：79.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

译者序

在朋友把这本书介绍给我的时候，我正在为一个项目中客户不断变化的需求而头痛不已，而实际上我的这个项目只是公司一大堆此类项目中的一个。

在现代的项目开发过程中，由于客户自身业务需求的快速变化而带来的对于软件需求上的变化越来越频繁。项目经理们所要面对的局面经常是固定的预算，固定的上线日期，加上频繁变化的需求这样的组合。这对于每一个精细业务定制软件的开发团队来说都是一个永恒的难题。当然从项目管理的角度我们需要通过客户管理和需求控制来避免这种情况的发生；但是另一方面，开发团队需要一套业务级别的实用软件框架来避免在客户需求发生变化的时候花时间重新搭建系统的业务平台，从而大大提高应对客户变化的能力以及自身的生产力。

坦白地讲，在此之前我对 ROCKFORD LHOTKA 的 CSLA.NET 知之甚少，甚至在开始的时候觉得这不过是一些底层代码的封装而已。然而在后来为本次翻译所作的必要准备、翻译的本身，以及后来实际的使用过程中，我对于这个业务框架的看法却经历了从不屑、接受，直到最后的推崇这样的变化。

CSLA.NET 最诱人之处就是它应用了.NET 中的很多特性，如远程访问、序列化、反射、企业服务、System.Transactions、强命名程序集、动态装载程序集和应用程序配置文件等，来在业务级别上真正地实现了构建绝大部分业务应用程序所需要的框架，而这个框架可以被轻松地应用在包括 Windows Forms、Web Forms 和 Web Services 在内的各种应用程序中，同时保持了相对来说非常高的性能。更重要的是，它还可以在不修改源代码的情况下，被部署在不同的硬件配置环境下，比如从所有代码都在同一台电脑上执行变成一个三层架构，要实现这一切，你所要做的只是修改一个配置文件而已。怎么样，是不是有相见恨晚的感觉了？

除了那些常见的底层功能代码封装之外，这个框架在业务级别上所支持的功能也异常强大，包括业务逻辑的 N 层撤销、移动对象、业务规则跟踪、业务授权规则、同一个对象拥有多种用户界面、Windows 和 Web Forms 的数据绑定，以及与分布式事务技术的集成等。有了这个框架，业务应用程序开发人员只需要考虑将精力集中在业务需求上，而完全不必要在那些对于增加应用程序的业务价值没有丝毫帮助的代码上面浪费时间。相信看到这里有很多项目经理和技术主管已经在回忆上一次与此相关的痛苦经历了，不过现在有了这么强大的框架摆在你的面前，你可以和这个难题说再见了。更重要的是，它完全是免费的！

这本 LHOTKA 的力作描述了怎样搭建支持这个架构的框架，并且展示了如何使用这个框架来创建基于业务对象的 Windows Forms、Web Forms 和 Web Services 应用程序。虽然这本书通篇的内容都是关于 LHOTKA 的宝贝框架 CSLA.NET 的介绍，但是通过他全面而平台性的讲解，我们也可以从中提炼出 CSLA.NET 2.0 架构当中的思想过程，同时把它当作一本讲述在.NET 平台上进行分布式架构的业务对象理论的书籍来参读。

随着微软 Visual Studio 2005 全系列产品的发布和.NET 2.0 的广泛应用，.NET 平台再次成为业务应用开发的焦点，特别是在业务逻辑频繁变化的行业领域，更是凭借其简短的学习路径，超高的生产力和灵活的应变能力而成为当仁不让的首选。而 CSLA.NET 则在这个计算平台上为我们提供了一个满足框架性业务操作的有力工具，有了它，我们在.NET 上的开发过程是不是有一种如虎添翼的感觉呢？

亲爱的读者朋友，希望这本书能对您的项目有所帮助，最后祝您编码顺利！

王鑫

2007 年 3 月

关于作者
About the Author

ROCKFORD LHOTKA 是一位著有大量书籍的作者，其中包括那本 *Expert VB 2005 Business Objects*。他是微软的地区总监，微软最有价值专家和 INETA 的发言人。ROCKFORD 在全世界无数的会议和用户组中发表演讲，并且他还是 MSDN 在线的一位专栏作者。除此之外，ROCKFORD 是 Magenic Technologies（www.magenic.com）的首席技术官，Magenic Technologies 是微软在美国最重要的金牌授权合作伙伴之一，致力于使用百分之百来自微软的工具和技术来解决当前最具挑战性的业务问题。

关于技术审阅者
About the Technical Reviewers

■**DAN BILLINGSLEY** 拥有数十年的专业软件开发的经验，他的工作遍及医疗、制造和服务行业。最近他还开始做一些系统和数据库管理及项目管理的工作。

他热心地支持和倡导 CSLA 和一般意义上的 N 层模型，总是积极地和他人分享他的经验和观点。

DAN 与他深爱的妻子和四个出色的孩子一起住在底特律市区。在除了家庭、教堂和工作以外的空闲时间里，他喜欢一头扎进视频游戏、彩弹球和侍弄花园当中，或者干脆去骑车远足。

■**BRANT ESTES** 是一位微软认证的解决方案开发人员和位于加州三藩市的 Magenic Tchnologies 的高级顾问。Brant 使用最新的.NET 技术为从平板电脑、移动设备到 Web 和 Windows 应用的众多应用程序进行了构架、设计并实现其技术解决方案。他非常享受萌生出全新和创新的方法来解决有趣的问题，并且喜欢紧跟最新的技术和小玩意。闲暇时，他喜欢吹他的小喇叭，招待客人，与他的猫玩耍，以及摄影。

当 Rocky 把他的第一版 CSLA 送到 Magenic 的时候，Brant 简直等不及 Rocky 把这个 VB 版本改写成 C#，所以他就自己承担了改写工作。那一天是周五。在周一的时候，Brant 已经改写了全部的超过一万行的 VB 代码。在后来的数月时间中，他花费了大量的时间在 CSLA 上面，成为了本书的技术审阅者，并且更重要的是，他成了 Rocky 的挚友。

■**PETAR KOZUL** 是一名 ComputerPro 的高级顾问，ComputerPro 是位于墨尔本的一家提供 IT 管理、咨询和企业解决方案的公司。他是 CSLA.NET 的一套扩展框架 ActiveObjects（http://csla.kozul.info）的作者。作为一名 CSLA 社区的活跃成员，他从 CSLA 刚出现的时候就开始使用这个框架了。他毕业于皇家墨尔本科技大学（RMIT），取得了计算机科学方面的学位。Petar 在软件设计和开发方面拥有超过 11 年的经验，主要集中在使用微软技术的面向对象的解决方案。他曾经在好几个国家工作过，包括克罗地亚、波黑和澳大利亚。他的工作涉及公共和私人领域的多个行业，包括游戏、零售、医药和政府部门。

鸣谢
Acknowledgments

本书开始的时候只是想作为一个修订本，但是后来到结束的时候几乎被完全重写了一遍，以便适用于.NET 2.0 和 Visual Studio 2005。这确实是一个很大的项目，因此，我想对那些帮助我写完这本书的人们表示感谢。

首先，我要感谢我的妻子和儿子们在过去的很多年里对我的爱，忍耐和支持。没有他们，这一切不可能实现。而且，我要特别感谢我的妻子帮助我完成编辑工作，因为她在写作过程中这一我最不喜欢的步骤里为我节省了大量的时间和工作。

我还要感谢 Greg Frankenfield 和 Paul Fridman，他们使 Magenic 成了一个很棒的工作乐园。他们和其他 Magenic 的员工给与我的支持太棒了，我对此非常感谢！很荣幸能和大家一起工作。

特别要感谢 Brant Estes，我在 Magenic 的一个好友，他帮我把初始的代码改写为 C#，并且在这几个月中一直把它与我的 VB 代码保持同步，由此节省了我无数的时间。谢谢你，Brant!

Magenic 的托管服务组织团队作了大量的测试工作，他们负责这个框架的很多单元测试。这个出色的团队帮我找到并改掉了很多缺陷，而且在保持 VB 和 C#代码同步方面扮演了关键角色。

感谢微软的 Steve Lasker 帮助我解决了一些在 Windows Forms 上的数据绑定问题，感谢 Bill McCarthy 帮助我把那些问题的解决方法打包进了 BindingSourceRefresh 控件。

Apress 的编辑们在本书上花了大量的时间和精力，这才使得本书成了你们今天看到的样子。我感谢他们所有人如此出色的工作。

最后，我要感谢许多给我写来电子邮件的人们，他们在信中支持我，鼓励我，或者只是问问我这本书什么时候能够完成。在这些书籍和 CSLA.NET 框架周围聚集起来的这个社区真是太棒了，我感谢你们所有人！我希望你们能在从阅读本书中得到收获，正像我从写作中得到的收获一样。

祝你们在编码中找到快乐！

本书简介
Introduction

这本书描述了怎样应用面向对象的概念来进行.NET 应用程序的架构、设计和开发。我把重点放在了面向业务的对象，即业务对象，以及怎样在包括 Web 和客户机/服务器结构的不同分布式环境中来实现它们。本书使用了大量的.NET 技术，面向对象的设计与编程思想，以及分布式的架构。

本书的前半部分叙述了如何在.NET 环境创建这个框架来支持面向对象的应用程序开发的流程。这其中包含了大量的架构概念和思想，还有一些比较深入的高级.NET 技术。

本书的后半部分应用这个框架创建了一个带有几个不同接口的示例应用程序。如果你愿意，你完全可以跳过本书的前半部分，直接使用这个框架来搭建面向对象的应用程序。

我创建这个 CSLA.NET 框架的一个主要目的就是为了简化.NET 开发。本书中使用该框架的开发人员无须关心底层的技术，如远程访问、序列化或反射的这些细节。所有的细节都被内建在框架当中，这使得使用框架的开发人员可以集中几乎所有的精力来关注业务逻辑和应用程序设计，而不是整天陷在那些琐碎的“管道连接”的问题里。

从.NET 1.0 到 2.0

本书是对上一版 *Expert C# Business Objects* 的一次主要修订。修订本主要加入了.NET 2.0 的新特性，以及应用了过去几年使用.NET 1.0 和 1.1 所学到的知识。

这本书几乎与 *Expert VB 2005 Business Objects* 那本书一模一样，唯一的区别就是这两本书使用的编程语言的语法不同。

这两本 VB 和 C#的书都体现了我在近十年的时间里研究的概念思想。我一贯的目标就是在分布式 N 层应用程序中实现高效的面向对象的设计。在这几年里，业界的技术、我对此的理解和对概念的表达都发生了极大的变化。

作为尝试分布式过程先驱的那些 VB5 和 VB6 的书籍描述了怎样使用 VB、COM、DCOM、MTS 和 COM+来创建面向对象的应用程序（或者至少它们尽可能地实现了用 VB5/6 和 COM 所能实现的面向对象）。它们还涉及分布式对象的概念，即给定的对象遍布在一个物理 N 层环境中的多台主机之间。在 COM 中，分布式对象很重要，所以这些书籍进行了大量的关于对象状态和状态序列化技术的讨论。

最后我写出了被我叫做 CSLA 的一个架构，意为基于组件的可扩展的逻辑架构。几年中，我

收到了上百封使用过 CSLA 的人给我发来的电子邮件，这些人把 CSLA 作为他们的应用程序的基础架构，这些程序中既有很小的单用户程序，也有支撑他们核心业务的大型企业应用。

在.NET 中，分布式对象的概念已经让位给了一个更好的想法——移动对象，就是说对象可以在一个 N 层环境中的计算机之间移动。在高级层面上来看，这与分布式对象是类似的，但是移动对象提供了一个在分布式环境中实现面向对象设计更强大的方法。

我还收到了相当多的电子邮件，说 CSLA.NET 不是很成功，这一点儿也不奇怪。要想有效地使用 CSLA.NET，你必须精通面向对象和基于组件的设计，理解分布式对象的概念，并且拥有其他大量的技能。移动对象架构有很多好处，但不是很好理解。

设计 CSLA.NET

.NET 的一个特点就是它经常提供多种方法来解决相同的问题。一些方法比另外一些要好，但是不一定马上就能找到给定问题的最好方法。在写那些.NET 1.0 的书的时候，我花了很长时间尝试着用不同的方法来实现分布式对象。尽管知道有多种方法能行得通，但是在最后我还是找到了最符合我最初目标的那种方法。

在我讨论那些目标之前，我想很有必要来谈谈另一个令我在写这本书的时候冥思苦想的问题。假设有很多的人使用了本书的上一个版本中的概念和代码，我想在任何可能的时候保留向后兼容的能力。同时，这本书的新版提供了一个好机会，来使得我们在应用.NET 2.0 的新特性的同时，应用过去几年中通过使用.NET 学到的东西。

应用学到的东西就是说要应用新的概念和代码，这需要修改已经存在的业务对象和用户界面代码。我是不接受弱向后兼容性的，然而发展这些概念来适应在面向对象及分布式计算领域的技术和我的观念这两方面的变化非常重要。

在可能的情况下，我会尽量地把对已经存在的代码的影响控制到最小，以便对于大多数应用程序来说，所需要的转换不会过于复杂。

我对架构和本书有一套详尽的目标。这些目标非常重要，因为它们对理解为什么我准备了许多种选择很关键，例如使用哪种.NET 技术和如何使用它们。这些目标有：

- 支持完全的面向对象编程模型
- 允许开发人员无障碍地使用架构
- 实现高可扩展性
- 实现高性能
- 提供CSLA旧版的所有功能和特性，如：
  - 为每个对象实现 N 层撤销（编辑、取消和应用）
  - 验证规则的管理
  - 授权规则的管理
  - 支持同一个对象拥有多种用户界面
  - 支持 Windows 和 Web Forms 的数据绑定
  - 与分布式事务技术的集成，如企业服务和 System.Transactions

- 通过处理如序列化、远程访问和反射这样复杂的问题，来简化.NET开发。
- 使用微软提供的工具，特别是Visual Studio.NET中的智能感应和自动完成。

在这其中，允许开发人员无障碍地使用架构，也就是说让开发人员“正常”地编程，对新版本的冲击可能最大。要想在没有框架的情况下达到这个目的，开发人员需要编写大量额外的代码来跟踪业务规则，实现 N 层撤销和支持对象数据的序列化。所有这些代码非常重要，但是对增加应用程序的业务价值没有丝毫的帮助。

幸运的是，.NET 提供了一些强大的技术来帮助减少或消除这样的“管道连接”代码。有了这些打包在框架里的技术，业务开发人员就根本不必为这些事情操心。简化代码这个目的有好几次决定了我如何设计架构。最后，这个框架在极大程度上实现了简化代码，开发人员只需要写出所需要的业务 C#类，然后就可以从框架中自动获得例如 N 层撤销和业务规则跟踪这样诸如此类的所有好处。

虽然花了我大量的时间和精力，但是我确实很高兴将这个架构写成本书，而且我希望你们能在开发自己的应用程序的时候发现它的好处。

本书范围

本书包含了 CSLA.NET 2.0 架构中的思想过程，描述了怎样搭建支持这个架构的框架，并且展示了如何使用这个框架来创建基于业务对象的 Windows Forms、Web Forms 和 Web Services 应用程序。

第 1 章介绍了分布式架构的一些概念，包括逻辑的架构和物理的架构、业务对象，以及分布式对象。或许更重要的是，这一章搭建好了一个舞台来展示本书后面其余部分的思想过程。

第 2 章继续使用了在第 1 章末尾描述的架构来开始搭建能实现之前提出的目标的框架。在第 2 章结束的时候，你将会看到对象的设计过程，这些对象会在第 4 章和第 5 章被实现出来；但是在此之前还有一些事情要做。

从第 3 章到第 5 章全都是关于 CSLA.NET 框架自身的创建。如果你对那些实现 N 层撤销、移动对象的支持、验证规则、授权规则，以及对象持久化的源代码感兴趣，你可以好好看看这些章节。此外，这些章节还用到了.NET 框架当中更高级和有趣的部分，包括远程访问、序列化、反射、.NET 安全、企业服务、System.Transactions、强命名程序集、动态装载程序集和应用程序配置文件等。

此后，本书的其他章节集中在如何使用这个架构和框架来搭建应用程序。即使你不想学习从第 3 章到第 5 章包含的所有底层的.NET 概念，你仍然可以阅读从第 6 章到第 12 章的这部分，使用这个框架来搭建基于这个框架的应用程序。

我在第 6 章中讨论了一个示例应用程序的需求，并创建了它的数据库。这个示例应用程序使用 SQL Server，还有表和存储过程来返回和更新数据。

第 7 章描述了怎样使用 CSLA.NET 中的每一个主要的基础类库来搭建你自己的业务对象。你会看到实现可编辑对象和只读对象、集合，以及名/值列表的基本代码结构。

第 8 章为应用程序创建业务对象。本章描述了你可以如何使用本框架来迅速容易地为一个应用程序搭建一套强大的业务对象。最终目的就是创建一套对象，来不仅对业务实体建模，而且提

供对 N 层撤销、数据绑定、能优化性能的多种物理配置、可扩展性、安全性，以及容错能力的支持，就像在第 1 章讨论的那样。

第 9 章演示了如何为业务对象创建 Windows Forms 接口。第 10 章讲述了用类似的功能创建 Web Forms 或 ASP.NET 接口。

第 11 章使用了 Web Services 来提供一个业务对象的编程接口，供 Web Services 客户端调用。

最后，第 12 章讲述了如何使用.NET Remoting、企业服务和 Web Services 来配置应用服务器。这些应用服务器支持 CSLA.NET 框架，并且可以互换地使用那些在从第 8 章到第 11 章创建的 Windows Forms、Web Forms 和 Web Services 应用程序。

在本书结束之前，你会得到一个实用的支持面向对象应用程序设计的框架。这个框架实现了一个可以在不同物理配置上部署的逻辑模型，来最好地支持 Windows、Web 和 Web Services 的客户端。

框架软件许可

许可与担保

8. 责任。不管是 Rockford Lhotka 还是任何本软件的贡献者都在法律允许的最大范围内不对与本软件及其许可有关的任何间接的、特殊的、结果的、意外的、惩罚性的或警戒性类型的损失负有责任，无论基于任何法律基础。你还必须在你以任何形式分发本软件的同时附上此有限责任。
9. 专利权。如果你对某人使用本软件提出专利权方面的诉讼，你的本软件使用许可自动终止。专利权，如果有的话，下文许可的只适用于本软件，而不适用于任何你的衍生作品。
10. 终止。如果你以任何方式违反了本许可，那么你被本许可赋予的权利自动终止。Rockford Lhotka 保留使用不同的许可条文来发布本软件，或随时中止分发本软件的权利。这样的选择权不会被用于收回本协议，而且本协议会继续保持全部效力，除非如上被终止。
11. 政府法律。本协议遵守美国明尼苏达州法律。
12. 无分配。本协议和其中的任何权益必须在得到开发人员明确的书面同意后才能进行分配。
13. 最终协议。本协议终止并代替之前所有的相关理解和协议。本协议可能在将来合适的时候由双方共同进行书面修改。
14. 服务条款。如果本协议中有任何条款被司法机构证明为无效或不能执行，那么本协议的其他条款将保持全部效力，就如那些无效或不能执行的条款从未被加入一样。
15. 标题。本协议中使用的标题只是为了方便，不能用于分析含义或意图。

使用本书所需的条件

本书中的代码已经用微软的 Visual Studio 2005 专业版，也就是.NET 框架 2.0 版测试过。数据库使用了包含在 Visual Studio 2005 专业版中的 SQL Server Express 版。Visual Studio 2005 企业版和完整版本的 SQL Server 非常有用，但是在这里不是必须的。

要想运行前面列举的工具和产品，你需要至少一台安装有 Windows 2000、Windows Server 2003 或 Windows XP 专业版的 PC。为了测试 CSLA.NET 对多物理层的支持，你需要为你想要添加的每一层准备一台 PC（或者你也可以使用 Virtual PC 或其他类似的工具）。

约定

我在本书中使用了一些不同的文本风格和布局来区分各种信息。这里有所用到风格的一些例子和含义解释。

代码有不同的字体。代码段用等宽字体，如 `foreach`。如果是一段你可以在程序中运行的代码，它会是这个样子：

```
if (Thread.CurrentPrincipal.Identity.IsAuthenticated)
{
  pnlUser.Text = Thread.CurrentPrincipal.Identity.Name;
  EnableMenus();
}
```

有时你会看到一些混合风格的代码，如：

```
dgProjects.DataSource = ProjectList.GetProjectList();
DataBind();

// Set security
System.Security.Principal.IPrincipal user;
user = Threading.Thread.CurrentPrincipal;
```

这个时候，普通字体用于你已经熟悉的，或者是不需要你马上动作的代码。粗体表示那些对比前面新添加的，或者我特别想让你关注的代码。

提示：这种风格表示建议、窍门和背景知识。

注意：这种风格表示重要信息。

编号段落采用的风格，如：

- 编号段落使用缩进的风格

如何下载本书代码

访问 Apress 网站（http://www.apress.com），用查找功能找到本书的书名。打开本书的详细页面，单击源代码链接。或者，在 Apress 主页的左手侧，单击源代码链接，在出现的文本框中选择本书。

下载文件是压缩了的，需要你用 WinZip 或 PKUnzip 这样的程序解压缩。代码一般是按照适合的目录结构组织好的，你只需要确保解压缩软件使用目录名字来解压文件就可以了。

作者和社区支持

本书和 CSLA.NET 框架拥有来自其作者和广大用户社区的支持。

本书作者维护着一个网站，该网站上可以找到常见问题及其答案、对框架的更新、一个在线论坛和额外的资源。社区的成员建立了额外的支持网站和工具，来帮助对 CSLA.NET 及其相关概念的理解和使用。

要想得到全部这些资源的信息和链接，请访问 http://www.lhotka.net/cslanet。

主要目录
Contents at a Glance

目录
Contents