实战.NET

——使用C#开发面向人的软件

Ronan Sorensen
George Shepherd
John Roberts
Russ Williams
著

英宇林琪侯战友 译

Pearson Education
清华大学出版社

实战 .NET

——使用 C#开发面向人的软件

Ronan Sorensen
George Shepherd
John Roberts
Russ Williams
著

英 宇 林 琪 侯战友 译

清华大学出版社

北京

EISBN: 0-201-73828-7

北京市版权局著作权合同登记号 图字 01-2002-4513 号

图书在版编目(CIP)数据

实战.NET——使用 C#开发面向人的软件/（美）索伦森等著；英宇，林琪，侯战友译. —北京：清华大学出版社，2003.7
书名原文：Applied .NET: Developing People-Oriented Software Using C#
ISBN 7-302-06431-8

Ⅰ.实… Ⅱ.①索… ②英… ③林… ④侯… Ⅲ.C 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CTP 数据核字（2003）第 018535 号

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
http://www.tup.com.cn
责任编辑：冯志强
印 刷 者：清华大学印刷厂
发 行 者：新华书店总店北京发行所
开 本：787×960 1/16 **印张**：24 **字数**：537 千字
版 次：2003 年 7 月第 1 版 2003 年 7 月第 1 次印刷
书 号：ISBN 7-302-06431-8/TP・4846
印 数：0001~4000
定 价：42.00 元

前　言

就像在海上积聚能量的风暴一样，新的计算时代的第一波浪潮已经开始撞击软件开发的“海岸”。这场风暴背后的力量已酝酿多时，随着它的登陆，有点猝不及防的软件业必须振作起来，并为在这场风暴中生存做好准备。

尽管采用上面的比喻来刻画目前计算机产业的状态有点夸张，但是这种比喻绝对是准确的。因为庞大的开发人员团体，不管他们是集中在一起的，还是分布合作的，还从来没有像现在这样面对如此众多的可用技术。就像一个站在海岸边的人，正面临着一场即将到来的风暴，作为一名开发人员，同样也能意识到软件产业正在经历一场变革。毫无疑问，你一定想努力领会这场变革的全部意义，并为此做出最佳的准备。希望本书能使你对这场变革获得一定程度的理解，帮助你在新的开发时代中生存下来，并能追赶潮流。本书提供的信息有助于读者理解 .NET 技术，它讲述了如何将所有这些技术融合在一起，迅速建立下一代的解决方案。本书涉及到的那些开发思想所使用的都是 C#，并通过几个 .NET 应用的开发来阐述。

《实战 .NET》(*Applied .NET*)从面向人的角度来观察促进软件开发变革的新力量，并为建立高效的 Internet 软件提供了一套开发原则。我们使用面向人(*people oriented*)这一术语来刻画即将到来的新的软件浪潮，因为该术语抓住了激发 .NET 风暴的原动力。该术语的起源要回溯到几年前 Ronan 于 1998 年所写的一本书——《Microsoft Windows NT Internet 开发技术内幕》。该书的第一部分介绍了崭新的面向人的编程模式，以及这类软件所具有的概念。第二部分阐述如何采用早期的技术来开发符合这些原则的系统。

.NET 技术为实现那本书中提出的目标和思想迈出了如此重要的一步，因而我们选择《使用 C#开发面向人的软件》作为本书副标题就是很自然的事情了。希望通过这个标题能够表达出如何采用 .NET 建立一种新的、令人激动的软件的看法。因此，尽管本书使用了 .NET 技术，但是建立这种应用的目标是为了创建更卓越的软件，这些软件被正式称作“面向人的软件”(people-oriented software)。

在《Microsoft Windows NT Internet 开发技术内幕》出版以来的这段时期，它所提出的思想已经趋于成熟，并成为本书作者所进行的各种讨论的焦点。其中一些讨论往往使问题更白热化，但是最后我们一致认为本书列出的原则都是正确的原则，未来的软件肯定将是面向人的。当 Microsoft 最近宣布了它的 HailStorm（冰雹）计划后，这一观点的论据显得

更加充分。对面向人的观点产生的任何怀疑都随之消失殆尽。一个崭新的时代正在到来，人们不必再面向计算机使用软件，软件将面向人们的实际生活。软件会运行在日常生活中的许多设备上，所有这些设备都会以不可思议的方式连接起来。在它们之中，核心主题是如何让软件嵌入社会，并面向于使用它们的人。

从面向人的角度出发， .NET 是通向最终目标的工具而不是最终的目标。显然除本书之外，会有更多的书详细介绍 .NET 的其他方面，并且为你提供有关方面的更大帮助。本书的重点则是在理论和实践之间寻求一个平衡点，它不但会介绍如何应用 .NET，而且还展示了这种应用能为你带来什么。我们认为《实战 .NET》包含的观点和原则给我们带来的好处远不止这些。本书会实实在在地向你展示如何应用所学的方法来建立真实的 .NET 应用——实践出真知。

致　　谢

作者要集体感谢以下人员：

- 感谢Addison-Wesley小组：Kristin Erickson，Curt Johnson，Chris Kief，Chanda Leary-Coutu，Marilyn Rash，Cathy Comer，Dianne Wood，Karin Hansen和Mark Bergeron等出版人员的建议，尤其要感谢我们敢作敢为的编辑Stephane Thomas。
- 感谢Plural全体同事的鼓励，特别要感谢James Watkins和Connie Hughes，他们为本书的改进提供了帮助。
- 感谢Miki Bell，他是本书的美工。
- 感谢Microsoft公司的Sanjay Parthasarathy，Nelson Rossa，Connie Sullivan和Rodney Miller提供的帮助和建议。
- 感谢Rob Howard，John McGuire，Greg Hack，Daryl Richter，Don Browning，Maxim Loukianov和Christophe Nasarre对本书做了技术审校。

此外，以下是每位作者分别的致谢。

首先，并且是最重要的，要感谢我的妻子Irene和三个女儿Mary、Catherine和Sophia对我写作这本书的支持。在这段时间里，她们知道我希望做什么，并且始终支持我的想法，鼓励我将这些想法写出来，因此我要特别感谢她们。我还要感谢我在爱尔兰、美国和意大利的亲戚，感谢他们给予的信任、希望和爱。最后，要感谢我的合作伙伴George、John和Russ感谢他们的友谊以及让我有机会与他们合作编写此书。

R.S.

我要特别感谢我的家人Sandy Daston和Ted Shepherd，在我辛苦写作本书时给予我的支持和问候。还要感谢DevelopMento为开发人员提供了大量练习和思考的场所。感谢Patrick Shepherd所作的大量宣传，他为现代软件筑起了另一面围栏（Remond所景仰的人是San Jose，而我则对Remond仰慕不已）。最后，要特别感谢Ronan Sorensen，John Roberts和Russ Williams，没有他们的努力，本书是不可能完成的。

G.S.

我想感谢Pete Nash，Mike Cabrera和Jason Cuplin给予的帮助和支持。感谢本书的合

作者，多年来与他们相识和共同工作使我得到了特别的快乐。感谢 Ronan Sorensen 在最近 4 年为我创造了许多很好的软件开发和出版的机会，当然包括这一次写作机会。感谢 Russ Williams 不断给予我的鼓励和灵感。感谢 George Shepherd 在 20 世纪 90 年代就鼓励我进行写作。与你们一起工作是我从事软件行业以来的重要部分。最后，我要特别感谢我的妻子 Sue，我的孩子 Daniel，Luke 以及 Michael 给予我的支持和理解。

J.R.

我首先要感谢我的家人在我写作本书期间做出的牺牲。因为我抽不出时间与他们共享闲暇时光，因而错过了棒球比赛，无法出席家庭聚会，并且带给他们疲惫的身心和心烦意乱的情绪，希望能得到他们的谅解。我要感谢我的女儿 Ryan，Chase 和 McKenzie 对我的理解，尤其是我的妻子 Gina 对我的大力支持，她在关键的时刻给予了我特别需要的鼓励。我有这样的妻子和孩子，是上帝给我的恩赐，我将永远感激不尽。我还要感谢本书的合著者，他们都曾与我共过事，我非常钦佩他们。能与他们合作编著本书是我的荣幸，在此感谢他们对提高本书的质量所做的贡献。尤其要感谢 Ronan，他做出了核心的贡献。正是他的辛勤工作、领导能力、经验、洞察力和潜质造就了你现在拥有的这本书。最后，我要感谢我的父母在这些年来对我的充分信任。在与我父亲的一次偶然交谈中，他讲述了自己对程序设计的偏爱，为我种下了从事软件开发的“种子”。这件事不仅是我职业选择的开始，而且还教育我与孩子们的任何交流都不能当成是小事。爸爸和妈妈，我非常地爱你们。

R.W.

尽管我们已经为避免本书出错付出了极大的努力，但是在印刷后难免会存在一些缺陷。Web 站点 http://www.people-oriented.net 会提供报告错误的详细联系方式，而且也会列出所有的勘误或更新列表。

目　录

第 1 章　面向人的软件

Internet 将软件引入到了大众中间。在这个世界上，一般的人都可以通过使用软件来相互联系，这在历史上尚属首次。随着 Internet 互连性扩展到了电视、广播、电话、个人数字助理(personal digital assistant, PDA)技术以及汽车等领域，这一趋势必将延续下去。除此之外，人们的日常生活也逐步成为软件所关注的重点，在此要么直接通过基于 Web 用户界面而实现的人类交互来完成，要么则是间接通过旨在满足人们需要的企业对企业(bussiness-to-business, B2B)的沟通来做到。一方面借助于软件使得互连性日渐增强，再加之软件更加侧重于以人为本，这就带来了软件设计的革新。

以往的软件所关注的是对某些事物操作建模，由此产生了面向对象思想的兴起。尽管在目前，“人”仍然可视作为一个面向对象世界中的对象集合，但是这种方法也许并不实用，而且很可能失败。利用面向对象设计，我们无法对这个社会中的各种动态交互和影响提供一种模糊的方法来建立模型。社会性的交互涉及到诸如自由度的使用、多种文化的引用、移动性、不可预见性以及地理位置等一系列问题，而以上也只是列出了相关问题中的一小部分。简单地说，仅利用一个对象模型的抽象是不足以充分表示一个社会的。人类的真实世界与事物世界大相径庭，正如哲学家 Karl Wojtyla（更为人熟知的是 John Paul II 主教）数年前所指出的：

> 我们所生活的这个世界由许多对象所组成……作为一个对象，人即为“某人”。这样就使其区别于这个可见世界中的其他实体，而一般来讲，一个对象往往仅作为“某物”。根据这一基本的区别，可以揭示出人类世界与事物世界之间所存在的鸿沟。[1]

由于人是类型迥异的对象，其行为就成为了软件开发的核心所在，由此就呈现出一种全新的编程方式，其目标为以一种更为专用的方法来处理人们之间复杂而动态的交互性。

.NET 平台即为这种编程方式的一个早期体现，它是面向人的。要充分了解 .NET 的诸多功能，很有必要对这种编程方式的各个元素加以理解。在人们从 C 语言的面向过程编程转向 C++语言的面向对象编程的那段时期，很容易产生以下的错误，即在对于设计用来解决问题的范例转换不甚了解之前，就试图草率地采用这一新工具。有些人错误地将 C++视

---

[1] Karol Wojtyla（John Paul II 主教），《Love and Responsibility（爱与责任）》。译者：H.T.Willetts（New York：William Collins Sons, London 和 Farrar, Straus 及 Giroux, 1981），21。在《爱与责任》一书中所提出的伦理准则可以普遍应用在 Internet 所产生影响的社会性方面。

为 C 的一种更好版本，而没有将它作为编写软件的一个完全不同的方法。同样地，当前的 .NET 也可能被错误地认为只是一种建立 Web 网站的更好的办法，而不是认为它能为面向下一代 Internet 提供支持技术。为了避免这种误解的产生，这一章余下的内容将对用于构建下一代 Internet 的面向人的编程范例加以探究，并讨论如何利用 .NET 平台实现这种范例。

Internet 将演化成什么，.NET 将如何提供帮助呢？Internet 的成功与人具有社会性这一事实有着不可分割的联系。我们很快就接纳了促进沟通的各种革新，从报纸、广播和电视的发展可见一斑。Internet 所提出的主要需求是人们需要参与到一个群体中——这是一个在线的群体，在规模上体现出全球性。这一全球性的在线群体很自然地被划分了大量更小的群体，这些小群体则以更专有的方式面向特定的团体。作为利用诸如 .NET 等技术而发展的下一代 Internet，它将通过经济、社会和文化交互与人做到更为全面也更加无缝的连接。

建立一个足以表示社会的在线群体是一项复杂的任务。尽管在速度和无线方面的显著进步对真正普及 Internet 是必要的，但工程师们面临的最大挑战却是要解决缺乏软件方法论这一问题。需要有更复杂且更强大的软件工具和技术来满足现有的大量工程任务。一旦有了合适的方法论和工具，一个全球性的在线群体即可从数以百万计的各自独立的人们中间脱颖而出。如若没有适当的工具、标准和方法论，那么进展将相当缓慢，而且会走很多弯路。

1.1 面向人的范例

要将万维网（World Wide Web, WWW）转换为一个更具有全球性的互连群体，所需的正确的软件方法论是什么呢？《Microsoft Windows NT Internet 开发技术内幕》[2]一书的第一部分介绍的一种面向人的编程范例可以解决这个问题。面向人的范例所强调的是通过 Internet 以更直接的方式在人们之间建立联系，另外该范例还强调将软件嵌入到社会运作之中，从而使在线群体得以产生。不同于软件行业以往所出现的诸如面向过程和面向对象编程等范例转换，面向人的编程并不关注于新的编程语言（如 Java）的创建。相反，面向人的编程所强调的是充分利用诸如 Windows Server 等现代操作系统所提供的丰富服务。

---

[2] Roman Sorensen, 《Inside Microsoft Windows NT Internet Development》(《Microsoft Windows NT Internet 开发技术内幕》)(Redmond,WA:Microsoft Press,1998)，5，未经授权不得使用。

1999 年 12 月，Pearson PTR、Slashdot.org、Netscape 公司的 DevEdge Online 以及 Doctor Dobb's Journal 向读者做出调查，要求就当今对计算机技术影响最大的书进行提名，也就是那些置于计算机旁并被翻得卷角了的书，而且它们常常可用于解决新的难题，这些书要经得起时间的考验，此外其价值将延续数年。《Microsoft Windows NT Internet 开发技术内幕》一书在 20 世纪的最佳计算机图书竞赛中位居第四。

就社会本身的行为方式而言，Internet 革命代表了一次重大的变革，由此对于软件的目标来讲，也势必带来巨大的转变，这是一种范例的转变，即将重点从原来强调单个计算任务的技术调整为强调社会交互、文化表述以及信息交换的技术。基本上，面向 Internet 设计的软件将负责建立一个全球性的群体。它将强调改善一般的生存环境，使得人们能更有效地完成日常活动。因此将这种新的范例称为面向人的编程(people oriented programming)一点也不为过。尽管 ActiveX 和 Java 为 Internet 的发展做出了卓越的贡献，它们本身却不足以满足 Internet 时代的需要。为了快速地建立可靠且可扩展的分布式软件解决方案，我们需要利用 ActiveX 和 Java，并将它们嵌入到支持 Internet 的系统中。这正是 Windows NT 服务器平台技术所提供的。Windows NT 和 Microsoft Windows 分布式互联网应用体系结构(Microsoft Windows Distributed interNet Applications Architecture，Windows DNA)就提供了用于实现面向人的编程的工具。[3]

这一指导原则最初在 Windows DNA 中实现，它在 Microsoft 的下一代 .NET 平台中得到了更全面的表述。 .NET 平台包含了 Windows DNA 的丰富的系统服务，另外还对其作了扩展从而得以创建面向人的 Web 服务，而这些服务能够以一种精心打造并具个性化的方式应用于 Internet 上。不过 Windows DNA 的重点在于满足类似 Internet 互连性、事务、异步编程、容错性、安全性和可扩展性等基本需求。 .NET 平台所解决的是面向人的 Web 服务需求。这些 Web 服务使人们将软件更加无缝地集成到他们的生活中去。例如，人们将能够以某种标准的方式查看他人的约会日历，或者是将其客户与供应商的业务处理加以集成。将重点放在人上面，在最近几年这一点对于软件行业的重要性日渐提高，而正因如此 .NET 平台应运而生。1999 年 3 月 29 日，Microsoft 宣布其公司已做彻底改造。Microsoft 力图通过卓越的软件提供一个全新而且应用更为广泛的方式来使人的能力得到增强，而不论何时、何地及在何种设备之上。Bill Gates，Microsoft 的主席和首席软件设计师，对于该公司更加以人为本的观点做出如下解释：

我们最初提出的“每家每户以及每个桌面上都有一台计算机”的观念仍然相当有意义。预视未来，这一观念将得到更大的发展。我们将看到一个新的世界，在这个世界中，人们可以使用任何计算设备在任何时间、任何地点完成所需做的任何工作。PC 仍将在这一未来世界中起到核心作用，但同时会有超乎想象的大量数字设备加入进来访问 Internet 的强大功能。我们希望为人们提供这种功能和互连性，以及选择如何在生活中进行计算的能力。[4]

.NET 平台将有助于创建更加面向人的软件，这是因为面向人的编程范例直接支持它所体现的 3 个概念的实现。这 3 个概念是（1）通用化、（2）协作和（3）转换。

---

[3] Roman Sorensen,《Inside Microsoft Windows NT Internet Development》,10.

[4] Microsoft press release(《Microsoft Announces Reorganization》,1999): http://www.microsoft.com/PressPass/features/1999/03-29reorg.asp.

1.1.1 通用化

通用化(Universalization)是一种开发模型，其依赖于复杂的通用运行时的能力，以实现普遍认可的 Internet 标准。

通用化模型依赖提供系统服务的运行时，这些系统服务可广泛应用于解决复杂的软件工程任务。尽管诸如 C 或 C++等编程语言也有着强大的功能，而且也提供运行时，但面向人的编程所提供的运行时在相应功能上则大为胜出。在此并没有把重点放在类似于封装、多态或继承等用于编写可重用代码的编程技术上，面向人的编程所强调的是对普遍存在的运行时所提供的服务进行重用，从而使代码创建工作最小化，同时代码编写也可以有所侧重，即更直接地为构建在线群体而开发面向人的 Web 服务。

编程的重点已由编程语言的内在功能转向了运行时的内在功能。Windows Server 即为实现了通用化 Internet 标准的运行时的典型例子，它在其组件对象模型(component object model, COM+)服务层中提供了极为丰富的特性。PocketPC 也是一个实现了通用化 Internet 标准的运行时例子，不过其服务有所缩减，从而适应于其所运行的设备。 .NET 运行时则更充分地体现了这一模型，对此将在本章稍后进行解释。

1.1.2 协作

协作(Collaboration)是一种合作模型，其中面向人的 Web 服务要相互合作以提供改进的服务。

协作模型有助于更为复杂的软件能够跨越组织界线实现集成。面向人的 Web 服务可以是任何在 Internet 上提供了可编程接口的应用，这与图形用户接口有所不同，前者的目的是为了使开发人员能够构建一个在线群体。在零售业中，此类 Web 服务的例子包括产品和附件目录、结账和支付处理服务以及配送和交付服务等。通过程序将这些 Web 服务捆绑在一起，全球的软件开发人员就可以联合起来创建一个大型“场所”，在此数百万的企业和消费者的需求都可得到识别和匹配。

要完成这一“壮举”，对软件工程来说可是相当艰难的挑战。在此需要利用目录来识别可用的 Web 服务，还需描述开发人员如何使之与其他服务集成。人们需要就 Web 服务描述约定达成共识以实现相同类型的服务，从而避免在集成多个服务提供商时复杂性过大。测试与故障诊断需要为编程人员提供简单易行的方法，从而在开发和操作阶段可以与多个 Web 服务提供商协作。

1.1.3 转换

转换(Translation)是一个互操作性模型，它所解决的是异构平台之间及不同服务描述约

定之间功能的转换。对于在一个存在颇多变化的环境中创建一个虚拟的联合体，转换模型提供了相应的方法。构建一个在线群体需要为数以千计的 Web 服务之间的相互通信提供无缝的方法。这绝不是一件容易的事，因为存在着数百万的软件开发人员使用着各种技术在独立地建立各类服务。Internet 是一个异构的基础设施，它可运行多种不同的操作系统，并可使用不兼容的组件协议，如分布式组件对象模型(distributed component object model, DCOM)、通用对象请求代理结构(Common Object Request Broker Architecture, CORBA)以及远程方法调用(remote method invocation, RMI)等。

将所有这些系统都迁移到一种通用技术并不可行。与此不同，面向人的编程所强调的是转换技术，从而使不同系统能够基于广泛应用的 Internet 标准实现通信，如超文本传输协议(Hypertext Transfer Protocol, HTTP)和可扩展标记语言(eXtensible Markup Language, XML)等。面向人的 Web 服务可以利用任何编程语言在任何平台上构建。在内部，服务可以使用专用协议来获得最大的可扩展性，而外部接口则可以在专用接口和普遍存在的 Internet 标准之间来回转换。转换模型还有另一个同等重要目标。多个 Web 服务提供者很可能提供同一类型的服务，但却使用不同的描述约定来指出如何通过程序来集成此类服务。软件开发人员若要试图适应所有这些不同的服务描述约定，这实在是一项相当复杂的任务。转换模型需要利用有关的工具和技术来实现这种方法，从而将所有这些描述约定都映像为消费系统能够理解的一个通用格式。

1.2　.NET 方法

为帮助软件工程人员建立面向人的系统，从而做到通用化、协作和转换性，.NET 平台在这方面可谓颇有建树的最早版本。尽管也可以使用另外的一些计算技术（如 Sun ONE）而不是 .NET 来实现这些原则，而且将来还将出现一些其他新技术，但由以下分析你可以了解到使用 .NET 平台的原因，下面这些分析的目的就是要提供一个示例，从而指出 .NET 如何有利于面向人的软件开发。本书余下部分则对此进行更为详尽的分析。

1.2.1　.NET 和通用化

相较于当前的 Internet，未来的 Internet 将更为普及，同时功能也更加强大，由此人们可以通过自然的接口彼此交互。Internet 将成为一个高带宽的全球性网络，它可以传输数据、声音和视频，并且能够连接遍布于世界各地的计算机、电话、广播、电视、PDA 和汽车等。

无线互连将相当快捷，同时价格也将可以承受，它很有可能取代有线电缆而成为 Internet 访问的最主要方式。新型 Internet 设备将纷至沓来，它们可以把诸如电冰箱、门、窗、空调器以及安全系统等家用设施连入到 Internet 中。到时候，可随意使用的 Internet 设

备将极为常见，人们甚至可以佩带、邮寄或者简单地更换这些设备。数据可以从任何地方访问，而且不会局限在单个 Internet 设备上。尽管安全可以利用生物技术得到改善，但是安全和隐私总是棘手的问题。使用 Internet 卡实现的金融事务会相当普及，而且在线购买与人工购买之间的界限也将变得模糊，因为这两者之间的主要区别是配送和选购。Internet 上将出现更为新颖、更加自然的通信方式。手写识别、语音识别以及通过摄像机的视觉识别将使人们忘记了一个复杂的技术基础设施的存在，而正是由此才使之可在 Web 上轻松地实现通信。可以想见，软件开发人员通过通用运行时的服务以及普遍接受的 Internet 标准，也能够忘却这些复杂性。

.NET 平台通过称为公共语言运行时的通用运行时将其触角进一步做了延伸。这一运行时要在操作系统的基础上进行操作，从而管理代码的执行并提供服务来简化开发过程。面向运行时的源代码称为托管代码(managed code)；编译器将其解释为一个 Microsoft 中间语言(Microsoft intermediate language, MSIL)，它独立于中央处理器(central processing unit, CPU)。在代码被执行时，即时(just-in-time, JIT)编译器会将此 MSIL 转换为其运行设备所需的特定于 CPU 的代码。从理论上说，这说明软件开发人员可以为运行时编写代码，而无需逐一针对其可能运行的每个 CPU 体系结构来开发。随着新的 Internet 设备逐渐建立在廉价的CPU产品基础之上，这一点的重要性也日益显现出来。尽管 .NET运行时对各种 Windows 操作系统版本均可用，但仍有可能将 .NET 移植到其他诸如 Linux 等操作系统上。为了更便于同其他平台上的各种通用运行时互操作， .NET 运行时利用了诸如 HTTP、HTTPS、XML 和简单对象访问协议(Simple Object Access Protocol, SOAP)等通用 Internet 标准来实现跨机通信服务。

.NET 运行时为目前在各种应用程序接口上可用的多种服务提供了一个统一的编程模型。当前存在着许多通用时，例如 WIN32 应用编程接口(application programming interface, API)，Microsoft 基类库(Microsoft Foundation Classes, MFC)、活动模板库(Active Template Library, ATL)以及 WinInet 等；还有一些更专用的库，如 DirectX、Microsoft 电话应用编程接口(Microsoft Telephony application programming interface, TAPI)和 CrypoAPI 等；此外还有大量用于组件服务的 COM 接口，如事务处理、排队组件或对象轮询等。要开发复杂的软件，以上都在必学之列。软件开发人员为了有效地利用诸如 Windows DNA 等体系结构，就需要吸收来源不同的诸多 API，其中有一些是相同功能的重复，有一些则面向不同的编程语言。 .NET 运行时基于一个简化的统一模型对这些 API 中的大多数做了实现，此模型抽象了许多细节，特别是可互操作的 COM 基础。将来，软件开发人员将能够把重点主要放在 .NET 运行时上，而不必考虑所用的是哪一种编程语言。例如， .NET 中的消息队列组件允许将基于消息的通信轻松地结合到某些应用中，这些应用需要完成诸如发送和接收消息、浏览现有队列或创建和删除队列等任务。HTTP 1.1 的运行时实现方式将开发人员从流水线操作、分块、加密、代理使用中解脱出来，此外也不必操心诸如 Basic、Kerberos 或 Windows NT Challenge/Response (NTLM)等证书和认证机制。

在 .NET 运行时中已经结合了许多强大的新特性来简化开发过程。其中包括通过一个公共类型系统(common type system, CTS)实现的跨语言集成、通过使用集合的版本简化和部署、通过可扩展元数据实现的自描述组件、利用自动垃圾回收实现的更简单的生命期管理、为实现组件交互而建立的一个简化模型以及改进的调试和配置服务等。

.NET 运行时引入了一个新的实体，称为应用程序域(AppDomain)，由此可以大大方便可扩展性设计，而这是连接上百万台设备时存在的主要问题。通常在高性能系统的设计中，可扩展性和容错性之间往往存在一场“拉锯战”。新组件被划分到一个应用的不同进程空间中，这样通过采用类似于内存访问调用等机制，即便存在未发现的漏洞，也不至于使整个系统陷入瘫痪状态。不过，跨进程通信将使应用的可扩展性显著下降，这是因为存在着必要的额外代码的执行，另外编组过程中在堆上进行的内存分配会造成处理器的串行化，这也是可扩展性下降的一个原因。在 .NET 体系结构中，托管代码得到保护，从而可以避免导致运行时中的一些典型错误。任何可能产生的负面结果都可以归结为“非法”应用程序域。这就使软件设计人员可以将代码执行划分到同一进程空间中的多个应用程序域，从而避免昂贵的进程间通信。最终结果即为一个更具可扩展性的系统，同时也可以获得容错。

安全性是确定 Internet 进化为一个复杂在线群体的另一个关键因素。除非人们确信其事务是安全的，否则他们将不会把敏感的业务操作连到 Web 上。 .NET 运行库提供了一个代码访问功能来帮助解决这一类问题。移动代码是一个相当大的威胁，因为这种代码可以取自于许多来源，例如在电子邮件的附件中，或者是从 Internet 下载的文档中。利用诸如 Internet 软件应用中的缓冲区溢出等已知的脆弱性是另一种常见的攻击方法。代码访问安全性可以帮助保护计算机系统免受此类攻击，因为它使得代码可以根据不同的级别授信，这要取决于代码来自哪里，以及其可能的用途。这种机制不会阻止所有移动代码不能执行，而只是对代码能够完成的工作加以限制。相应的信任程度取决于代码是否由一个可信任源做了数字签名。代码访问安全性还可减少其他合法软件被恶意代码通过缓冲区溢出或其他漏洞而滥用的风险。这是通过指定合法代码允许执行的一组操作和决不允许它执行的一组操作来实现的。

.NET 运行时是一个仍在发展的平台，随着对逐渐浮出水面的在线群体的需求不断增长，它还将继续得到改进。我们完全可以预料到，通用运行时的以后版本必将结合诸如自然接口等额外的特性。这些新的特性可以通过同一个统一编程模型提供。

1.2.2　.NET 和协作

较之于具体实现，在理论上构建一个全球互连的在线群体可能要容易得多。而实现时所需的协作水平却很难达到，特别是当所涉及的不同行业在给定的竞争活力条件下更是如此。不过，当今现有 Internet 的存在表明：当通过协作而产生的机会较之于保护专有利益而得到的优势要更胜一筹时，取得共识是有可能的。在下一代 Internet 中可能需要的是何

种类型的应用呢？

这是一个很大的题目，因此我们只将重点放在一个电子商务的例子上。假设每种产品和服务都有一个全球性的惟一识别码。你在商厦或超市所购买的每一样东西都有相应的识别码，可以用你家里、车上或 PDA 上方便可用的扫描设备进行扫描，所有这些都可连入 Internet。另外还可假设存在面向人的 Web 服务，它可以理解你的身份，并提供个性化存储。无论你何时购买，也不论购买何种商品，相应商品都可得到扫描，而你的个人记录也会得到更新。任何时候消费了某个商品，你都可以对其扫描，相应地你的个人记录将会反映出来。在为你的记录设置个性化参数后，自动软件代理将周期性地检查数据以确保你的家庭供应总能得到补充。如果啤酒快喝完了，自动软件代理就会向当地商店再订购一些，这样在你下次进该商店时就可以顺便取回来。还有一种可能，即自动代理可以基于此商品的惟一识别码来搜索当前价格，并从一个更廉价而且送货上门的商店里订购一些额外商品。每个月自动代理都会为你所购买的商品生成一个报告，同时提供可能对你有好处的不同购买模式的对照分析。例如，代理可能会告诉你购买 X 牌而不是 Y 牌更省钱，或者是如果你把产品 X 在家存放上两个月，那么通过批发购买更省钱。每个月自动代理都会为你的账户提供一个财务总结，并询问是否允许自动付账。

产品的每个供应商也可能有其相应的自动代理。通过分析以前的购买模式，供应商可以更为准确地预测到其商品的需要程度。这可用于从生产商那里自动订购额外的供应产品，而生产商相应地也可依靠自动代理来适当地处理原材料的库存。如果存在不可预知的事件打破了传统的购买模式，此时就能够对生产迅速地做出相应调整，过量的供应可以打折提供给自动代理。如此的最终结果是一个得到调优的系统，它可以使浪费最小化，并且有效地保证供求平衡。

.NET 如何帮助软件开发人员构建这样的系统呢？仅当存在多家公司的服务通过 Web 实现程序化协作的方法时，构建这种系统才是有可能的。另外还需要开发大量的 Web 服务，从而以良定义的方式提供业务功能。通过使用普遍接受的标准，.NET 平台极大地简化了这种 Web 服务的创建。运行时提供了所有必要的信息管道(plumbing)，而且诸如 Visual Studio.NET 等开发工具也提供了创建 Web 服务应用构架的先导，利用特定功能还可以对其加以改进。.NET 平台通过一组名称空间提供了这些功能，在此将提及其中的一部分。

System.Web.Services 名称空间包括了支持用户建立和使用 Web 服务的类。使用这一功能相当容易。例如，为了使一个公共类的某方法（运行于 ASP.NET 中）可以通过 Internet 访问，用户只需将 WebMethod 属性增加到其定义即可。System.Web.Services.Protocols 名称空间所包括的类可定义用于通过电缆来传输数据的协议，这里数据传输发生在 Web 服务客户和 Web 服务本身之间的通信过程中。它提供了诸如 HttpClientRequest 和 HttpServerResponse 等方法，而且提供了通过 HTTP 与 SOAP Web 服务进行通信的实现方式。

System.Web.Services.Description 名称空间包括了支持利用某种服务描述语言公开描述

一个 Web 服务的类。服务描述约定是当 Web 服务在 Visual Studio 中创建时自动生成的。Web 服务的消费者使用此约定来学习如何与之通信，也就是说，所约定的是它可以调用的方法、输入参数以及可能返回的响应的确切格式。对于以此方式来描述 Web 服务，Web 服务描述语言(Web services description language, WSDL)已经成为了用于此目的的事实上的 XML Internet 标准。System.Web.Services.Discovery 名称空间包括了允许消费者定位可用的 Web 服务的类。Web 服务发现是一个了解可用 Web 服务存在的进程，另外还可查询它的描述约定，以便用户能够正确地与之交互。

通用发现、描述和集成(Universal Discovery, Description, and Integration, UDDI)项目(http://www.uddi.org)的创建是力图通过一个分布式的 Web 服务目录为 Web 服务的集成提供一个框架。此目录允许用户在某个行业或某个特定公司中定位可用的 Web 服务。 .NET 平台还支持利用预定义 SOAP 消息的分类进行 Web 服务的程序性注册和发现。UDDI 和 WSDL 标准是建立在 IBM、Microsoft 以及 Ariba 的协作基础上出现的，而且不少于 30 家的其他软件公司也对此认可。

许多 Web 服务是通过绑定和扩展其他 Web 服务而建立的。有些本质上讲算是面向人的 Web 服务很可能作为“日常服务”的形式出现。数据存储 Web 服务可能允许人们在一个通用的可访问位置上安全地存储其信息。访问相应数据的方法需要考虑到用户的 Internet 连接的速度，而且应该允许离线进行更新从而在稍后再进行同步。尽管数据可以作为 XML 来存储，用户还应该能够使用 Microsoft Office 应用程序来查看并对之进行修改。另外需要利用身份 Web 服务来允许对多个 Internet 服务实现单点登录。Microsoft Passport 服务目前就提供这种功能。诸如生物特性等更复杂的特性可能随后会增加进来。用户将需要通知和消息等 Web 服务，它们向人们或软件代理发送信息，如股票价格更新或新闻标题等。在线日历服务也是需要的，它可以使人们或代理通过协作安排约会和会议。日历服务还应支持用户设置授权许可，从而对访问进行授权或禁止。例如，可能允许一个用户访问重要的客户，但拒绝他访问未获邀请的销售会议。 .NET 平台和相关的 Web 服务都是尚在发展的系统，它们总是在不断地得到更新和改进。一个动态交付服务将很可能出现，它允许用户在出现改进时自动接收这些改进。

将 Web 服务绑定在一起，将会对软件开发群体提出一些很有意思的问题。软件开发过程将演化为包含协作开发等特殊要求。对于结合了多个活动系统（而且对这些系统还无从控制）的应用，开发人员如何对其进行测试和调试呢？有可能将它们与诊断信息结合并进行有效的管理吗？如何预防预料之外的情况呢？比如某个顾客在一个死循环中重复调用一个 Web 服务而导致拒绝服务的产生；事务又如何跨越多个 Web 服务，从而在出现错误时能够自动将相应的更改撤销呢？尽管 .NET 平台是为有利于协作开发所迈出的第一步，但仍有许多问题亟待解决。

1.2.3 .NET 和转换

直至 2001 年 8 月， .NET 平台仍处于测试版本。尽管 Internet 已经连接了世界各地数百万计的计算机，但其中仅有 1%的极少数在运行 .NET 软件。Web 上计算机之间的通信非常少，而且主要局限于信息的表示和点击式购销操作。即便是诸如 XML、SOAP、WSDL 和 UDDI 等标准已经出台，以便于协作式的 B2B 通信，但目前可用的系统实现并不很多。

现在作为 Internet 的主要绑定“粘合剂”的仍为 HTML，它已经应用于诸如电子商务等非表示性任务中。HTML 获得了巨大成功，这是由于它很易于实现，而且对于存在于大量异构 Internet 环境中的许多不同类型的系统，它是相当通用的转换语言。尽管实际上连到 Internet 的每台计算机都能够理解像传输控制协议 Internet 协议(Transmission Control Protocol/Internet Protocol, TCP/IP)和 HTTP 等 Internet 协议，但这些协议却限制了通信水平以及其间可能出现的协作程度。

解决这一问题的一个方法可能是让每个人都接受一种通用的技术或诸如 C#等可以提供更好服务的新编程语言。不过就目前的软件行业来说，这种方法并不实际，而且它会扼制将来进一步的革新，因此也并不令人满意。采用一种转换方法则可以提供更好的灵活性和表述的自由度。

如果 Web 要发展为一个更具有全球性的相连的在线群体，那么就需要新的转换机制来支持人们之间更为丰富且更为复杂的软件交互。只有普遍接受的协议是远远不够的，用户还需要相应的平台和工具来简化其实现。 .NET 平台将有助于解决在线群体的转换需求，在此有两种主要方法：系统互操作性和服务约定转换。

1. 系统互操作性

尽管 Internet 通信仍然相当基本，不过许多复杂的业务处理已经实现了计算机化。目前大量的软件工程方法业已存在，它们可用于构造一个在线群体。但其中大多数均锁定于公司内部，这是因为它们使用了专用且种类不同的软件协议。 .NET 平台可以帮助公司解除对这些丰富资源的限制，从而把它们提供到 Web 上。软件开发人员可以为现有系统构建中间 Web 服务作为转换层，还可以将某些传统代码输出到 .NET。 .NET 平台中的公共类型系统 CTS 有助于更简单的代码转换，这是通过处理不同语言所使用的不兼容的一般数据类型、跨语言的集成功能以及处理事件、动态行为、持久性、属性和异常等的标准化方法而实现的。Web 服务的 .NET 实现方式是基于 XML 的，可以由任何语言、组件模型或操作系统访问，这是因为它没有绑定于某个特定的组件技术或对象调用规则。这就说明多个不同公司可以独立地构建 Web 服务，而这些 Web 服务可以实现互操作而无需在开始时对系统级实现方式的细节取得共识。如果用户已经有了一个基于 CORBA 或 COM 的系统，他们就能够建立一个 .NET Web 转换服务，此服务将包装其功能并在 Web 上提供。 .NET 体系结构使用了简单的可扩展 SOAP 协议，从而在 Internet 的异构条件下实现信息交换。

SOAP 是一个基于 XML 的协议，它并没有定义任何应用或实现语义，而且可以应用到从异步消息到远程过程调用等各种系统之中。

另外，.NET 平台中的 Web 表单也简化了 HTML 的生成，它相当于一个用于表示信息的转换机制。使用 Web 表单，用户可以通过在一个设计环境中“拖拉”丰富的用户界面控件来创建 Web 页面，再增加有关代码，从而用任何编程语言通过程序将这些控件绑定到业务层。.NET 平台将把所需的表示转换为纯 HTML，而任何设备和操作系统上的浏览器都能够理解 HTML。

2. 约定转换

对于希望利用协作在 Web 上开展业务的公司来说，所面临的最大难题将是如何就特定行业的 Web 服务描述约定达成一致。将会出现多个约定，它们都起到相同的用途，但在格式上却有稍许差别。例如，可能会有数以百计的订购约定类型；如果没有合适的转换技术，程序员将试图对每一种订购约定都进行处理，这实在是一场梦魇。为了简化这种错综复杂性，用户需要有一种简便的方法将不同的 Web 服务约定转换为使用 Web 服务时所希望的一个通用格式。幸运的是，可扩展样式语言转换(extensible style language transformation, XSLT)规范即以一种标准化方式解决了这一问题。XSLT 是一种将 XML 文档类型转换为其他 XML 文档类型的语言。在 XSLT 中表述的转换描述了将一个源树转换为一个结果树的规则，它是通过将模式与模板关联来实现的。.NET 平台利用 System.Xml 名称空间支持的类实现了 XML 文档对象模型(document object model, DOM)，它还将 XML DOM 与 ADO.NET 所提供的数据访问服务做了统一。.NET 的 System.Xml.Xsl 名称空间实现了万维网联盟(World Wide Web Consortium, W3C)XSLT 规范。XslTransform 类可以使用 XmlReader 加载一个 XSL 样式表，而且可以用 XmlNavigator 对输入数据进行转换。

对于系统互操作性以及服务约定转换，.NET 平台为之提供了简便的转换机制，这一功能是使 Web 发展为一个复杂的在线群体所迈出的一大步。可以想象，这些功能还将出现在其他技术和平台中，从而使所有 Internet 系统都可以通过更为强大的方式轻松地集成。

1.3 小　结

Internet 已经改变了软件开发的规则，而且面向人的软件范例初现端倪，其目标是将 Internet 转换为一个足以表示社会的在线群体。通用化、协作和转换是面向人的范例所体现的三个原则。Web 向一个复杂在线群体的转换已经开始了，对于本章介绍的面向人的范例特征，许多开发人员业已在构建体现出这些特征的软件。不过，当前开发人员是通过手工来完成所有结构性工作的，同时实现一种基于个案基础的 Internet 标准。.NET 框架准备了各种必要的工具来使软件的通用化、协作和转换更易于交付，从而更有效地将 Web 转换为一个真正意义上的全球性在线群体。

第 2 章　应用面向人的软件

第 1 章归纳了设计面向人的软件范例的三个原则，同时探究了采用 .NET 平台实现这类软件的适用性。运用范例的价值在于它将一个统一主题的若干个概念综合起来。这就像一个过滤器，我们可以通过它来观察现实世界，把握问题的关键以及亟须引起注意的范围。范例为那些看起来复杂的、随意的、多层面的事物提供了结构和次序。正确理解一个范例是非常关键的，因为它给如何思考问题提供了模型或思维框架。

本章介绍如何将面向人的程序设计原则应用到 InternetBaton Web Collaboration（InternetBaton Web 协作）应用的设计与开发中，相应的站点位于 http://www.internetbaton.com。InternetBaton 是一种崭新的应用，它是完全以 .NET 框架为基础编码而构建的。

2.1　面向人的设计

随着 Internet 逐步转变为一种面向全球的在线社区，面向人的软件原则也随之得以发展。虽然比特和字节仍很重要，但是人们关注的焦点已经转移到如何将软件融入社会的日常活动中，进而通过它促进人类的交流。面向人绝非忽视软件已经具备的特点，而是促使它进一步地改进，并立足于一种特殊的视角指导它的发展。随着面向对象原则的采用，一个新的世界正逐步引起人们的普遍关注。在这个新世界中，不管是直接地通过人机交互界面，还是间接地通过企业对企业(B2B)的通信，人已经成为接受服务的主要“对象”。

同样，面向人的设计方法论应该利用面向对象设计中最好的原则。好的软件体系结构不仅要始终保持完整性，还要具有一致的表现，才能确保它是一种成功的设计模式。综合的软件体系结构和设计通常强调这些常见主题：

- 目标适宜度：根据特性集合和功能，将系统正确地映射到应用的业务需求上。
- 性能：无论怎样安排系统的使用方式，都要确保系统能满足所需的吞吐量、响应时间和执行时间等度量标准。
- 可伸缩性：不管是由于吞吐量还是并发访问用户数量的增加或者扩大，系统都能方便地满足不断增长的性能需求。
- 安全性：防止系统受到攻击破坏，并且在身份认证、访问授权、不可否认性、保密性、数据完整性和审计跟踪方面提供完备的保护手段。

- 私有性：确保系统能满足相应行业类型中规定的隐私保护条款。
- 可靠性：采用了完备的错误处理技术，正确地遵循了事务处理的原子性、一致性、独立性和持久性原则。
- 可用性：通过使用冗余或者集群手段，在体系结构中整合进容错技术，使任何错误都能得到屏蔽，从而不会导致系统运行失败。
- 可管理性：将系统管理集中起来，使用仪器和监控技术将可视化集成到管理操作中。
- 可维护性：能够预见到由于业务需求的变化或者新技术的出现而引发的系统改进，并且能便利地实现这种改进。

人类型:“基于态度的设计”

为了在现有的软件设计过程中对通用化、协作和转换原则投入额外的关注，人们采用了一种称为“设计态度”的面向人的设计理念。经验证明，人的因素肯定会影响软件项目开发成果的成功。软件体系结构明显是由人构成的，因此它的结构能从正反两个方面影响并促进软件的设计。态度为人们采用一种特殊的思考方式提供了活力和动机。设计态度方法论通过识别和关注给定项目任务的最佳精神层面，寻求利用人类的这种自然本性。为了增强采用这种方法的优势，我们仅考虑工作环境，因为工作环境清楚地揭示出态度决定人们工作表现的良好程度。此外，某些特定的态度往往被组合在一起，共存于相同类型的人中。例如，成功的部门管理员具有这样一种态度倾向，通过某种政策激励下属，使他们具有完成新要求的热情。“人类型”这个术语用来概括某种类型的人所共同具有的一组确定态度，这些态度能相互促进并达到平衡。正是由于某种类型的人所具有的不同态度能够彼此相互影响，才形成他们固有的行为特点。

在业务环境中，通过人类型能将那些在其职业挑战中取得成功的人的典型态度明确下来。例如，注重细节和做事谨慎这种态度和特点，是成为一名会计师所必需的品质。同样，人类型能帮助软件开发的架构师确定应该使用哪些态度才能成功地完成某类项目。比如，一名架构师在设计一个协作软件的认证机制时，会采用“安全警卫”这类人的态度。“安全警卫”类型的人的态度包括谨慎、充满怀疑，以及为合法用户提供可靠的访问途径的义务。重要的是我们要注意到人类型常常与要被解决的特定问题相关。在设计虚拟导游程序的软件结构时，肯定不适合采用“安全警卫”类型的人的态度。

每种类型的人都明确具有一组设计态度。我们可以使用每个特殊的类型来更详细、更精确地描述设计态度。设计态度方法论成为一项面向人的软件开发原则，并且产生了三种不同类型的人：矿工、指挥家和语言学家。

1. 矿工

当我们考虑通用化所涉及到的因素时，就出现了“矿工”这一人类型，他们具有特别

适合于挖掘潜在珍贵资源的设计态度。矿工的观点可以归纳为如下特征：

- 开采，而不是建设。
- 致力于发现。
- 当发现资源不足时，能很快地减少损失。
- 通过额外的技术革新，有选择地开采某些难以开发的资源。
- 承认用户终究想继续改进最新的技术。

2. 指挥家

当考虑协作方面的因素时，“指挥家”这种人类型具有特别适合于支持这项原则的设计态度。指挥家的观点可以归纳为如下特征：

- 致力于采用集体的方式领导其他人一起完成任务。
- 能意识到整体发挥的力量大于局部力量的总和。
- 接受这样的事实：每个演员需要依靠其他人才能达到预期的效果。
- 有替补演员能替换缺席的演员。
- 为了达到像交响乐那样的效果，能反复排练要表演的内容。

3. 语言学家

当考虑转换所涉及到的因素时，“语言学家”这一人类型具有特别适合于支持该原则的设计态度。语言学家的观点可以归纳为如下特征：

- 接受表达方式往往具有多样性的事实。
- 理解不同的成分能构成更丰富、更多样化的环境。
- 注重表达的真实含义和来源。
- 预见到新的表达方式会不断被发现和产生。
- 实用 Pareto 原则：大约有 20%的可选表达方式能满足 80%的使用需要。

这三种人类型能帮助软件架构师以一种崭新的方式思考如何使用面向人的模型建立软件工程。这些类型的人所具有的见解是被专门挑选出来的，它们主要能在通用化、协作和转换等重要方面发挥作用，并且能突出表达与每个原则相适应的部署和观点。但是，人类型所具有的特点，就像印刷铅版一样，不能适用于各种环境。例如，具有爱尔兰传统观念的人爱好写作，并喜欢喝啤酒。尽管他们的特点与本章的作者相符，因为作者是出生并成长于爱尔兰的，但是很多爱尔兰人对这两点都不喜欢。

同样，可能存在这样的情况，在相应的人类型中存在一项特殊态度，它无法适用于某个设计环境或者业务需求。例如，一个编程人员可能因为版权问题而不能选择采用 .NET 服务器，如商业服务器(Commerce Server)提供的丰富特性。由此提出这样一个观点：在具有某种人类型所有特点的程序中严格采用精神层面的设计原则是不必要的，或者是无益的。使用相应的人类型的主要目的是为了形成最佳的面向人的设计部署。显然，在设计 InternetBaton 应用时遵循了这项原则。

设计态度的后继介绍

为了讲述 InternetBaton 应用的设计过程，我们在这里只是简单介绍了设计态度的概念。关于设计态度这一主题的完整介绍将贯穿于整本书中，它会为每一方面的设计，如性能、可扩展性、安全性、保密性、可靠性、可用性、可管理性和可维护性概括出相应的人类型。

设想一下，通过与软件产业中顶尖的架构师的交流开发出的这些“人类型”是多么有用。这些架构师在软件行业中共享他们在设计态度方面累积的知识，实践证明这些知识在软件构造的各个方面的应用都是成功的。以这些经验为基础，工程设计人员在解决特殊的软件设计难题时，能探讨是否可以采用人类型所表达的各种设计态度。

2.2　概念运用：InternetBaton 应用

InternetBaton 应用是一个分散的 Web 协作应用，它能够让人们实时地工作于某个项目，尤其是当协作人员属于不同的组织或者位于不同的地理位置时。有效的协作需要提供这样一种机制：通过它，人们可以采用辅助手段更新或者改进位于共享空间中的共同资源。InternetBaton 提供了这种共享空间，并且克服了身份认证、访问授权、并发访问和安全编辑共享空间中共同资源的复杂性。

2.2.1　InternetBaton 应用特征

尽管已经存在许多类似的应用，InternetBaton 应用还是有自己的特色，因为它的共享资源并没有存放在 InternetBaton 的 Web 服务器上。相反，它存储的是指向共享资源的虚拟统一资源定位器(URL：Uniform Resource Locator)。以共享文档为例，我们设想本书的作者们想合作编写该书的前言部分。首先要达成这一共识：指向前言的 URL 是 http://www.InternetBaton.com/me@email.com.Book/perface.bt。当我们访问该 URL 时，会被重定向到最新的前言版本上，它可能被存放在 Internet 上的任何位置。例如，我们中间有人会将文档存放在家用 PC 机上，该机器通过一条数字用户线路(DSL：Digital Subscriber Line)始终与 Internet 保持着连接，而其他作者可能将文档保存在由各种 Internet 服务提供商(ISP：Internet Service Provider)维护的 Web 服务器上。如果最新的前言版本不在我的家用 PC 机上，如果我想要更新它，那么可以在浏览器中输入 URL：http://www.InternetBaton.com/me@email.com.Book/perface.bt，就能被自动重定向到文档的真正位置。

如果文档的状态显示它还没有登出(check out)，那么我就可以登出它，并将其复制到

自己的家用 PC 机的共享区中，同样，该计算机通过 DSL 始终与 Internet 保持着连接。完成文档的编辑以后，通过更新虚拟的 URL：http://www.InternetBaton.com/me@email.com.Book/ perface.bt，就可以登入(check in)文档，使该 URL 指向我的家用 PC Web 服务器，然后其他作者就能被导向到我更新过的最新版本。你可能会问自己为什么不能简单地将文档复制到某个作者同事的家用 PC 机上。其中一个原因可能是他不容许其他人通过 Internet 向他的计算机上载文件，这能帮他预防某些人在他的计算机上植入病毒或者特洛伊木马程序。另一个好处是，他能保留自己的原始版本，直到他决定采用我的新版本时才更新它。而且，如果我们当中某位作者的 PC 机硬盘被损坏了，并且该作者的文档是最新的版本，那么我们只会损失他所做的修改部分。

在其他许多情况中，采用分布式的体系结构具有更多的优点。例如，共享敏感文档的商业组织想要完全控制这些文档的存储位置。它们不想把这些文档放置在 ISP 管理的共享区上，而更愿意把它们存放在于自己管理的 Web 服务器上，并且使用具有数字证书的公共密钥基础设施来授权对它们的访问。可以使用 InternetBaton 来管理公司之间的协作要求，其中任何一家公司都不用将真正的文档存放在 InternetBaton 的 Web 站点上。采用这种方式，需要提供一个共享空间，它具有管理、通知和跟踪的特性，却不会危及共享资源的实际安全。如果本书的作者特别想要保护对前言的访问，他们可将虚拟的 URL：http://www.InternetBaton.com/me@email.com.Book/perface.bt 指向类似于 https://www.a-very-secured-web-server/Book/Perface 的 URL。当访问这个 URL 时，会弹出一个对话框，要求输入用户 ID 和密码，但是却不用在 InternetBaton 的 Web 站点上共享这些资源。

当某个人不得不与其他未知的或者完全不信任的人打交道时，通过这个例子能证明使用类似于 InternetBaton 这样的分布式存储体系结构能带来很大的好处。有成千上万的人愿意免费为某个项目工作，比如将某本书翻译成各种语言。项目经理可以让文档在更新它的自愿者之间交流。InternetBaton 通过虚拟 URL 维护着文档的当前位置，以及一个 E-mail 地址列表和对文档做出过修改的所有人员的共享位置。如果翻译结果的质量不错，经过周期性的时间间隔，项目经理会将文档的主拷贝更新为最近的版本。否则会将文档恢复为前一个版本，并提供给更多的人翻译。你或许会问为什么项目经理不是简单地将文档存放于每个人均可以更新的一个位置。他之所以不想这样做是因为任何个人都可能会轻易地破坏文档的内容，因此所做的全部修改都会丢失。如果采用分布式的 InternetBaton 应用体系结构就不会发生这样的情况，因为用户只能修改他们从共享位置下载下来的文档版本。如果用户破坏了位于自己的共享位置上的文档，或者将虚拟的 URL 指向了一个虚假的位置，项目经理能轻易地将虚拟 URL 复位为合法的位置。

为什么使用 InternetBaton 这个名字?

对这个应用来说，baton（指挥棒）这个单词具有双重含义。其一是指指挥交响乐队的指挥家所用的细木棒。按照这个意思，它用于表示 Web 上的协作应用，因为虚拟的 URL 就像指挥家的指挥棒，它能引起每个人的注意，采用它能有效管理每个参与人员所做出的共同努力。但是 baton 也可以表示接力队员所使用的中间为空的圆柱型木棍。它由一个队员递交给下一个队员。InternetBaton 的应用能形象地比喻为接力赛，因为每个小组成员只跑接力赛的一段，当另一个队员跑时，他不会跟着跑。将接力棒交给小组的下一个接力成员意味着下一个人要继续完成下一段赛跑。在 InternetBaton 应用中，一次只有一个人能修改资源。只有当虚拟 URL 被定向到一个新的位置时，才可以把资源提供给另一个成员使用。

2.2.2　通用化设计：挖掘运行时

既然已经理解了 InternetBaton 应用的特点，就可以继续从通用化的角度探讨它的设计与开发。需要采用矿工人类型的设计态度，用来代表“开采，而不是建设”和“致力于发现”等方面的特长。不难想象在 .NET 通用运行时中有大量丰富的资源有待挖掘，使用这些资源能帮助实现 InternetBaton 应用（如表 2-1 所示）。

表 2-1　用于 InternetBaton 实现的 .NET 资源

步　骤	InternetBaton 功能	所用到的 .NET 框架元素
1	带登录验证的注册页	Web 应用、验证控件和数据库的插入
2	对用户的认证和授权	ASP.NET 运行时环境和数据库查询的认证和授权元素
3	创建新的 Baton 项目	数据库更新、DataGrid 控件以及带定制事件的用户控件
4	InternetBaton 与其他 Web 服务的集成	Web 服务和定制的 HTTP 处理程序
5	Baton 的同步	异步的 Web 服务调用
6	Baton 元数据的转换	可扩展的标记语言转换

第一步：带登录验证的注册页

第一步是使用 Visual Studio.NET 创建一个新的 Web 应用，称之为 SignupBaton。这一步要完成的是一项简单的任务，因为向导会帮助你生成一个名为 WebForm1.aspx 的文件，可以在其中添加注册页面所需的元素。为了添加标签、按钮和文本框，只要使用所见即所