Designing Scalable .NET Applications

设计可扩展的 .NET 应用程序

- 描述使用 Microsoft 技术构建的可扩展.NET 应用程序的体系结构
- 全面阐述适用于 IT 架构师、系统设计师和开发人员的可扩展性设计
- 强调正确设计对于避免可扩展性问题的重要性

(瑞典) Joachim Rossberg 著
Rickard Redler
卞 军 周 磊 译

清华大学出版社

设计可扩展的.NET 应用程序

(瑞典) Joachim Rossberg
Rickard Redler 著

卞军 周磊 译

清华大学出版社

北 京

内 容 简 介

本书揭示了 Windows Server 2003 和 Microsoft .NET 平台提供的最新功能，从概念到部署全面讲述了正确设计.NET 企业应用程序的完整过程，包括 UML 建模、数据库设计和实现、选择合适的操作系统、设计基础结构，以及编写和部署代码等内容。

本书适用于 IT 架构师、系统设计师和开发人员。

EISBN: 1-59059-214-X
Designing Scalable .NET Applications
Joachim Rossberg　Rickard Redler
Original English language edition published by Apress L. P., 2560 Ninth Street, Suite 219, Berkeley, CA 94710 USA.

北京市版权局著作权合同登记号　图字：01-2004-2051

图书在版编目(CIP)数据

设计可扩展的.NET 应用程序/(美)罗斯伯格(Rossberg, J.)，(美)瑞德勒(Redler, R.)著；卞军，周磊译. —北京：清华大学出版社，2005.7
书名原文：Designing Scalable .NET Applications
ISBN 7-302-11114-6

Ⅰ. 设… Ⅱ.①罗… ②瑞… ③卞…④周… Ⅲ.计算机网络—程序设计 Ⅳ. TP393
中国版本图书馆 CIP 数据核字(2005)第 054265 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　**客户服务**：010-62776969
组稿编辑：曹　康
文稿编辑：于　平
封面设计：康　博
版式设计：康　博
印 装 者：三河市春园印刷有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：22.75　**字数**：582 千字
版　　次：2005 年 7 月第 1 版　2005 年 7 月第1 次印刷
书　　号：ISBN 7-302-11114-6/TP · 7351
印　　数：1 ~ 4000
定　　价：48.00 元

前　言

我们发现很多设计师和系统架构师缺乏对如何使用 Microsoft 技术构建并实现大型企业解决方案的理解。架构师们在考虑基于这种技术构建重要任务系统时经常犹豫不决——并不是因为他们已经尝试并失败了，而仅仅是因为他们没有很好地理解应该使用哪种工具来构建。我们期望能够改变这种情况。

从 2002 年起我们就有编写本书的想法。开始考虑将这本书编写成 Cap Gemini Ernst & Young 公司的内部文档。但当进行市场调研的时候，我们发现很少有针对 IT 架构师和系统设计师的书籍。大部分书籍都是针对开发人员的，我们需要一本面向更广泛读者的书。因为我们考虑很多 IT 架构师缺乏对用 Microsoft 平台可以执行什么任务的全面理解，所以决定应该将文档的读者扩展到 Cap Gemini Ernst & Young 公司之外，并试图将它作为图书出版，于是就和 Apress 合作出版了这本书。

本书读者对象

本书读者对象主要是设计师和 IT 架构师，但是我们尽量覆盖到那些觉得对开发人员也有用的主题。首先，让我们定义这三种读者类型的人。不同的公司对这些术语可能有不同的定义，所以为了避免混淆，在这里阐述清楚我们的定义。

1. 架构师

架构师需要和客户(或者决策者)一起考虑应用程序或者解决方案的需求以及数据流。架构师还要定义使应用程序工作的服务器，功能块等。架构师和公司的管理层一起，在较高的层面上研究如何设计应用程序。他们同样还要考虑和其他系统的集成。

架构师不用具备很深的技术技能；他们在抽象的层面上进行设计，并给出解决方案的蓝图。架构师可以是应用程序架构师和基础结构架构师。基础结构架构师关注连网问题——应该如何配置集群，以及如何保护基础结构。应用程序架构师关心应用程序以及应用程序的设计。最好的结果是这两类架构师在设计阶段紧密协作。

2. 设计师

设计师从架构师处获取蓝图，然后完成基于诸如 Microsoft 或者 Java 等平台的设计。

这类人可能是有领域内丰富经验的开发人员。相对于架构师，他们对技术非常在行。

也有设计师设计基础结构。这些人决定使用何种集群解决方案，实现何种服务器版本，使用何种安全技术等。

3. 开发人员

最后我们介绍开发人员，他们负责实现设计师给出的应用程序设计。这些“编码者”完成最后的工作。但是开发人员也可以是实现架构师和设计师决定使用的基础结构的人。

本书结构

我们希望本书读者在使用本书时能够获得灵感。本书的目的是介绍 Microsoft 的技术，同样也说明如今您可以在 Microsoft 平台上构建应用程序。

本书简要描述了企业应用程序的重要部分，并介绍了我们多年来积累的关于设计应用程序的经验。设计总是一个成功项目的关键。

因为本书着重于设计，所以不会深入讲解企业应用程序的细节。相反，本书试图在架构师/设计师和开发人员之间搭建一座桥梁。

注意：

即使本书主要倾向于 Microsoft 的方法，但是如果认为这是惟一的技术那就错了。当非 Microsoft 的技术可以向客户提供更好的解决方案时，我们自己都会使用其他技术，如 Java 和 Linux。

接下来是本书的各章内容简介。

第 1 章：企业应用程序设计简介

本章大致介绍一些和企业应用程序设计与实现相关的重要话题。概述了企业应用程序集成(EAI)、统一建模语言(UML)和对象角色建模(ORM)。

第 2 章：操作系统与.NET 企业服务器

本章介绍 Microsoft 可用的软件种类，使用它们可以为应用程序构建好的平台。本章讨论了操作系统和.NET 企业服务器，并研究它们如何适合企业应用程序的设计。

第 3 章：集群技术

本章概述了 Windows 提供的两种集群服务器的技术。网络负载平衡(Network Load Balancing，NLB)和 Microsoft 集群服务(Microsoft Cluster Service，MSCS)被集成到 Windows Server 系列中，并用于增强可扩展性、可靠性和有效性。

本章还仔细介绍了 Application Center，它是.NET 企业服务器。这种服务器协助您以简单的方式管理集群。

第 4 章：Windows Server 系列概览

本章比较深入地介绍了 Windows Server 操作系统，展示了 Windows 体系结构，以及应该如何在平台上使用 NLB 和 MSCS。本章还讨论了 Windows 中的安全性。

第 5 章：企业应用程序的体系结构

第 5 章介绍企业应用程序本身。本章讨论了如何设计企业应用程序，以及其他一些在设计阶段很关键的话题。

第 6 章：Web 服务设计和实践

每个人都听说过 Web 服务。如果没有，或者仅仅只想知道更多的内容，那么就要阅读这一章。本章讨论了设计和安全话题，同时还讨论了何时应该使用 Web 服务，何时应该使用.NET Remoting 代替 Web 服务。

第 7 章：Internet Information Service

本章分析了 Internet Information Service(IIS)。本章描述了它的体系结构，如何使用 ASP.NET，以及如何调整和保护 IIS。

第 8 章：数据存储设计和 SQL Server

数据存储在所有企业中都很重要。需要一个考虑周全的存储策略，这样才可以降低开销，并提高工作效率。这里描述了可以如何通过设计合适的数据存储来合并数据。

本章中还讨论了 SQL Server 的体系结构、性能和安全。

第 9 章：示例应用程序

本章描述了如何通过本书提供的技巧和方法来设计企业应用程序，给出的示例应用程序是一个用于大型企业的时间报告应用程序。

本书涉及的内容

本书将描述很多我们在实际工作中积累的有关设计企业应用程序的无价经验。过去一些年来，越来越多的客户请求集成解决方案。通过介绍 SOAP 和 XML，我们发现可以使用以前设计这些应用程序的一些想法。当然，随着新技术的引入，我们一直在改进自己的想法，并修改设计模式。Sten Sundblad 和 Per Sundblad 是我们灵感和知识的重要来源之一，他们是 *Designing for Scalability Using Windows DNA*(Microsoft Press，2000。ISBN：0-735-60968-3)及其续篇 *Design Patterns for Scalable Microsoft .NET Application*(由他们自己的公司出版，并可在 http://www.2xsundblad.com 获取)的作者。建议访问他们的网站，以获取更多关于设计模式的知识。

本书未涉及的内容

在本书中我们将不会讨论集成本身。相反，我们着重于可以用很多解决方案实现的一般设计，不管它们的目的是什么。相对于其他书，我们将尽量覆盖更广泛的内容。如果我们已经了解了某项知识，将它加入到本书就很重要。如果一个大型项目需要及时发布，并且同时要实现所有的期望，那么开发人员和设计师都需要对它有明确的认识。很明显，这意味着在很多话题上，我们不能想多深入就多深入，但是幸运的是其他的书做到了这一点。比如，前面提到的 Sten Sundblad 和 Per Sundblad 的书对设计模式和建模建议很有价值。其他一些书提供了对操作系统、数据库、Web 服务，XML 和其他对我们很重要的领域的深入讨论。本书试图在上述技术之间构筑桥梁，以便您能够为客户或者公司构建更好的应用程序。

构建企业应用程序并不是一件轻松的事。如果从开始时没有正确的设计，失败的风险将会剧增。不良的设计可能不会马上发现，但是随着时间的流逝，一定会出现性能问题和扩展问题。为了避免这些，IT 架构师和系统设计师需要具备关于何种技术有效以及如何使用它们的知识。

本书的目的是作为学习更多关于本书主题的向导。我们相信读者会发现本书对于您的职业生涯将大有裨益。我们希望您喜欢阅读本书。

从网站 www.tupwk.com.cn 可以免费下载本书的源代码。

目　录

第 1 章　企业应用程序设计简介

当今的所有公司都很重视各种形式的应用程序和数据。这种情况已经持续了很久一段时间，而且随着现代企业的全球化，应用程序和数据将会变得越来越重要。人类开始群体生活时，信息就是社会的一个重要组成部分。那时，人们需要密切关注季节的变化，决定何时打猎、播种以及进行其他活动。当人类社会变得越来越复杂时，人们对信息的需求也越来越复杂。

本章的开始将探讨一个问题：因为企业的信息分布在多个方面，所以企业经常难于检索有用的信息。接着引入一个概念：企业应用程序集成(Enterprise Application Integration, EAI)，这个概念的出现解决了上述问题。

尽管在普通多层应用程序中，集成可能会引入不必要的技术和工具，但是它不会改变太多的基础设计，因此程序开发人员能够继续使用他们已有的知识。

需要指出的是，和其他项目一样，可扩展性的设计对于集成项目来说也一样重要。在企业应用程序设计中必须考虑可扩展性的设计。不必要的延迟和差性能花费时间和金钱，如果在设计中不考虑可扩展性的设计，即使应用程序部署完成，也很可能是一个失败的应用程序。编写本书就是为了让读者知道一些技巧，以构建更成功的扩展性好的解决方案。

提示：

从多年的实际工作经验可以获知，应该将应用程序设计成可扩展的。尽管对于一个特殊的应用程序，开始的设计目的只是被少数人使用，但是大多数的应用程序都有一种发展趋势：被成千上万的人使用。如果在应用程序开发的最初阶段考虑了可扩展性的设计，在今后的使用过程中就不必考虑这种发展趋势问题。只需分离不同的层，将它们放置在不同的计算机或者集群上。这样不需要重新改写应用程序，就可以让它为多个用户服务。

1.1　回顾

在 80 年代，华尔街的股票交易人使用独立的终端。很明显，如果把终端、资产交易数据和管理系统连接到一起，就可以方便地纵览所有的经纪业务信息，作出更好的决策。这种需求可能促成了“应用程序集成”观念的诞生。之后，许多其他业务也有了相似的需求。

集成的需求成为业务关注的重心，原因是多方面的。比如，客户机/服务器的解决方案开始代替主服务器系统后，应用程序的开发人员突然需要为其公司构建更加灵活的解决方案。当时，许多解决方案和应用程序都是公司的私有财产。当两个公司合并时，不同的环境使得数据映射很困难。这就需要降低原有应用程序的复杂度。为了让这些应用程序协同工作，一种解决方法就是应用程序集成。90 年代由于 Internet 的爆炸式发展，许多公司采用新方法来开展自身业务。企业到企业(Business-to-Business，B2B)和企业到客户(Business-to-Consumer, B2C)的商机不可避免地导致了集成的需要，不止是企业之间的集成，还有企业和客户之间的集成。

第一代 EAI 解决方案的结构有两种：集线器和辐条(Hub-and-Spoke)结构，面向集成的总线(Integration-Bus-Oriented)结构。一种逻辑结构的分离首先开始出现在应用程序中，比如，数据

和传输的分离，请求和响应的分离，发行人和订阅者的分离，等。接着引入了 CORBA 和 COM 标准。人们开始讨论松散耦合的通信。但是 CORBA 和 COM 存在一个问题，它们依然是私有的。很难集成这两种标准。同时上述解决方法中还存在一些性能和可扩展性的问题。

业务的持续发展不断促成了 EAI 的改变。现在 EAI 这个术语也包括消息代理、业务进程流管理和工作流。企业要降低成本，简化需要集成的应用程序的创建、管理和修改就变得越来越重要。改变应用程序时，企业不能总是依赖技术人员和开发人员，同时也需要重用业务逻辑，来削减成本和降低复杂度。幸运的是，已经开发的诸如 XML、SOAP 和 Web 服务之类的标准，使得集成更加简单、有效和安全。

1.2 现代企业

假设现在有一个公司，R&R 汽车公司(R&R Automobile Corporation)，它生产和销售汽车。这个公司已经存在了许多年。公司内部有成千上万的数据和通信系统，没有人知道精确的数目。这些通信系统驻留在不同的硬件和软件平台上；可能 70%的系统驻留在主服务器上，其余分布在客户机/服务器和 PC 机系统中。不同时间内数据在不同系统之间传递。一些系统实时地传递和接收数据，而另外一些系统成批的传递和接收数据，一天一次甚至一周一次。少数系统完全不能和其他系统通信，R&R 汽车公司不得不打印这些系统的信息，然后把这些信息手工输入到其他系统中。

体系结构的混乱使得 R&R 公司很难知道公司与客户之间关系的细节(见图 1-1)。这明显是公司的一个问题，尤其是在公司必须对业务环境的变化要作出及时反应的情况下。

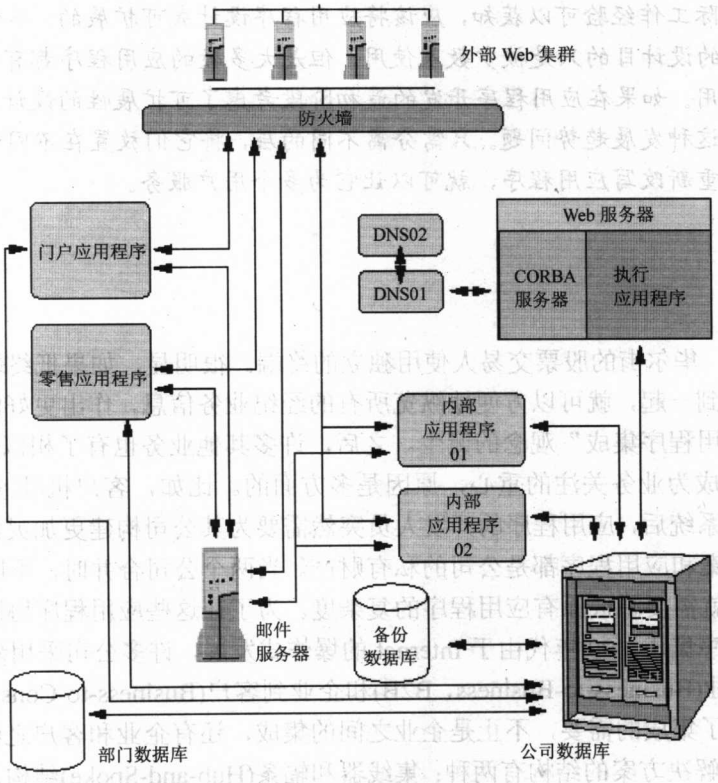

图 1-1 R&R 汽车公司中混乱的体系结构

R&R 管理部门发现与合伙人和客户要有更密切的交互。这就是说，公司需要得到反馈的供应链、客户支持及服务的信息，这样 R&R 公司才能成为一个高效的现代公司。这种交互可以通过提供应用程序之间更好的集成来实现。但是实现集成存在一些障碍。在 R&R 公司运作的这些年中，在不同的部门之间、地理上分散的办公室之间，以及生产线的不同环节之间，诸如此类，都已经确立了边界，而这种边界会降低公司内部信息共享的能力。容易理解，如果公司的这种障碍很大，客户就很难得到实惠。然而，企业的集成将提供有效的契机，呈现给客户统一的界面。集成还可以提供更有效的终端到终端(End-to-End)进程。

现在看看 R&R 公司的集成团队所面临的一些问题，也可以说是一些挑战：

- **R&R 使用了多种技术**。在过去几年中，使用了各种技术来构建 R&R 的应用程序和数据库。如前所述，一些系统提供了对外部的接口，另一些系统则没有提供。公司充斥着客户关系管理(CRM，Customer Relationship Management)系统与其他应用程序，它们是标准的或者是私有的。这些应用程序通常运行在不同的平台上。
- **R&R 的新应用程序需要扩展传统系统提供的特性**。更新传统系统的成本非常巨大，甚至不如更换系统。而且一些系统几乎不能更改。
- **需要开发新的业务工作流**。R&R 需要找到方案，为其业务进程开发新的工作流。但是已有的应用程序需要集成到新开发的工作流中。
- **许多正在使用的应用程序，其业务逻辑、用户接口和数据访问之间没有清晰的界限**。在一个 n 层的现代应用程序中，设计者和开发人员共同构建这个应用程序，让业务逻辑、用户接口和数据访问等项目分离，如图 1-2 所示。

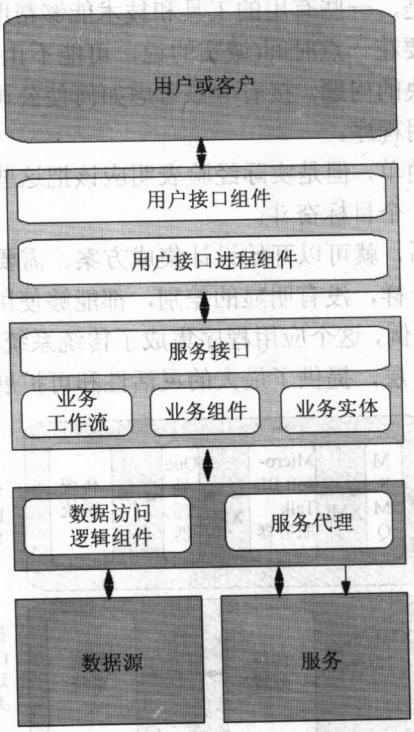

图 1-2　一个 n 层的应用程序模型

- **应用程序之间传递的数据有不同的格式**。使用 EDI 文档、文本文件与 XML 文件。因此很难把一种数据结构的字段映射到另一种数据结构的字段。
- **使用不同的组件模型**。一些应用程序使用 CORBA，而另一些应用程序使用 COM/COM+。这种局面导致紧密耦合的应用程序相互之间不能方便地通信。
- **R&R 内部的系统不能与外界集成**。尽管这种情况在公司内部不会出现问题，但是当 R&R 与客户及合伙人交互时，就会出现很多问题。附带也产生一个问题，R&R 公司不能控制客户及合伙人的其他系统，正如 R&R 不能控制自身系统一样。这说明 R&R 的集成团队需要调整与外界通信的方式。
- **大多数传统系统运行良好**。改变一个传统系统来提供新的特性，会引入故障和错误的风险。是否值得花费时间和金钱来做这些改变，是一个需要思索的问题。同时为了实现这种改变，集成团队需要一些有关传统系统的文档，通读这些文档会让开发人员对传统系统有更好的理解，但是这些文档通常是不存在的。
- **现在部署的集成方案必须是开放式的，这样能够支持今后的基础结构需求**。要遵循这个原则，必须在设计方案中引入一些理念。R&R 的集成团队需要仔细考虑公司正在使用的技术，尽可能地遵循已有的标准。这样 R&R 团队在选择新技术时就不会选择一些具有迷惑性的最新技术，至少不会选择相对未知或者未经证实的技术。

综上所述，对于 R&R 的集成团队来说，集成并不是一件容易的事情。“集成”本身可以看作一个拼图，所有的碎片都分散放在地上。团队的首要工作就是用这些碎片构建一个完整的图片。这是“集成”希望奋斗实现的幻想。幻想可能永远都不能实现，但是如果目标足够远大，就有可能贴近幻想。幸运的是，一些有用的工具和技术能够帮助实现目标。

在开始实施集成之前，要花一点时间(诚实的说，可能不止是一点时间)来考虑一些问题。首先必须弄清楚实际需要解决的问题。接着应该考虑如何使公司获得的利润最大。然后开始指定需要构建与需要连接的应用程序。

这些问题看起来似乎很简单，但是实际经验表明应该把这些问题的答案形成文档，这样可以确保每个开发人员都朝着一个目标奋斗。

有了这些问题的答案之后，就可以开始设计集成方案。需要说明的是，设计企业的集成项目与设计其他企业应用程序一样，没有明显的差别，都能够使用同样的设计模式。

图 1-3 是一个应用程序示例，这个应用程序集成了传统系统的数据和新的功能。如图所示，这个应用程序依然被设计成 n 层，提供了最大的灵活性和可扩展性。

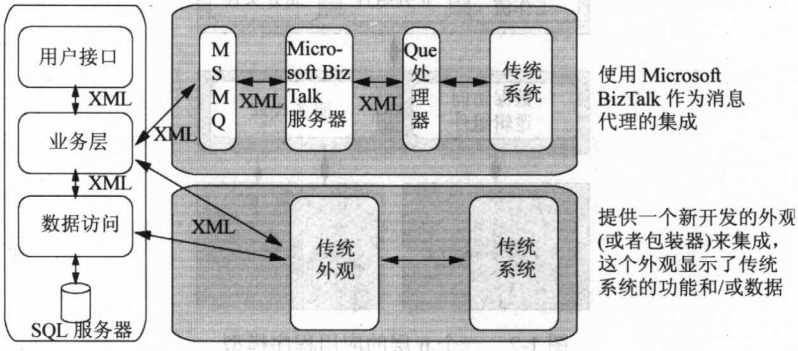

图 1-3 一个集成了传统系统的 n 层应用程序模型