内附双CD

系统架构与开发

《程序员》杂志社 曾登高 编著

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

.NET 系统架构与开发

《程序员》杂志社 曾登高 编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书系统而详尽地介绍了.NET 基本概念，多层架构的实现，开发流程中测试、部署、移植、安全、性能以及团队开发的实施步骤。第一次让读者从概念到开发、从开发到部署全面了解.NET 系统架构。

本书适合于有一定.NET概念急需提高的软件开发者、需要深入了解.NET系统架构的程序架构设计师阅读。

随书所附光盘收录了由微软公司平台策略部门经理 Robert Hess 主持的.NET SHOW 系统讲座的精选内容（10小时以上英文访谈，中文字幕），可以帮助开发人员更好地理解.NET 框架的设计理念和有关的实现技术。

图书在版编目（CIP）数据

.NET 系统架构与开发／曾登高编著. —北京：电子工业出版社，2003.4

ISBN 7-5053-8656-5

Ⅰ.N... Ⅱ.曾... Ⅲ.计算机网络－程序设计 Ⅳ.TP393

中国版本图书馆 CIP 数据核字（2003）第 026985 号

责任编辑：孟迎霞
特邀编辑：林宇彦
印　　刷：北京中科印刷有限公司
出版发行：电子工业出版社　http://www.phei.com.cn
　　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：170×230　1/16　印张：46.5　字数：1210千字　附光盘2张
版　　次：2003年4月第1版　2003年4月第1次印刷
印　　数：6000册　　定价：69.00元（2CD）

前 言

欢迎阅读《.NET 系统架构与开发》一书，本书旨在帮助开发人员在.NET 平台上有效地进行应用程序开发。

.NET 框架是一种新的计算平台，它在各个方面简化了在分布式 Internet 环境中的应用程序开发。崭新的开发工具 Visual Studio.NET 的推出，更是使许多曾经让人望而生畏的编程任务变得简单、明了。可以这样说，.NET 正深刻地影响着现有的开发模式和方法。

和 Java 的理想目标“在多种平台上使用一种语言”不同，.NET 架构致力于解决“多种语言共享一种平台”。目前，VB、VC、Delphi、Python、COBOL 等语言都有了.NET 的版本，随着技术的发展，开发人员可以在.NET 平台上运用更多的语言来实现应用。可以想像，未来 10 年的开发领域，.NET 框架将显得越来越重要。为此，你做好准备了吗？

读者对象

本书主要面向那些已经入门.NET 的开发者和项目管理者。如果你打算在.NET 平台上开发应用程序，或者你想进一步了解.NET 架构和技术，你都应该读一读这本书，它会给你许多有益的建议。

本书并不是讲述某种编程语言知识的教材，在阅读本书前，读者应该熟悉 Visual Basic 或者 c# 语言，了解.NET 平台的有关特性。书中的所有文章都由多位具有 10 年以上开发经验的微软专家撰写，技术讲解由浅入深，对程序的正确开发、部署和

管理均具有很好的指导作用。

本书第一章阐述了.NET 系统架构的术语、模式、概念和定义，初学者或者是有经验的.NET 开发人员都应该读一下这些内容，它可以为你奠定更坚实的基础。第二章介绍了数据层、逻辑层和表现层的相关实现技术，如果你对.NET 有了基本的了解，阅读这部分内容可以使你深入地了解各种技术的实现细节和技巧。第三章对程序调试、团队开发、部属、性能和安全问题进行了专题讨论，如果你正在开发一个项目，这部分肯定会给你提供很好的帮助和建议。

本书的目标

.NET 方面的图书在市面上已经非常多了，大部分图书只讲解单个领域或语言的应用，适合初学者循序渐进地学习.NET 程序开发。但是，相对于.NET 系统架构、部署、性能、安全、团队开发管理等话题涉及的则非常少。

本书应该作为在.NET 平台上开发应用程序的指南。该书的宗旨不是讲解基本的编程知识，它围绕着.NET 系统架构对各种编程方案、编程技巧进行了详细的论述和比较。参照书中的有关指导，你可以有效地分析当前所面临的问题，并做出最好的抉择，比如：部署方案的选择，事务处理模型的选择。

为问题提供解决方案

软件本身在飞快地发展，同时构建这些应用程序所允许的时间在飞速地减少。经常的情况是：一方面，开发人员想更好地了解技术以使他们能更好地使用技术；另一方面，却没有足够的时间来准备项目所需的经验和技巧。

上面情形的结果可能是：开发者需要摸着石头过河。如果他比较幸运的话，最终会得到一个可行的应用程序，虽然设计上可能由于知识不完备而显得散乱，但至少它是可用的。如果不幸运，项目可能会失败，其原因主要是缺乏足够的信息做出正确的抉择。

本书给出了构建.NET 应用程序过程中碰到的典型问题的解

决方案。通过阅读本书，你将能针对自己项目的特殊需要提出更合理的设计。

提供解决问题的背景知识和相关文档

开发者必须知道怎样将一种特殊的方法运用到自己所需解决的问题中，本书各章节不但提供了解决方案的源代码，而且还详细分析了为什么要这么做以及这些方案是如何起作用的。

碰到了问题，怎么办？MSDN 一般是最佳的选择。可面对庞大的MSDN，你也许会觉得无从下手。本书以各种常见问题为主线，在各章节中为主题提供了许多的链接。一方面，你可以通过这些链接了解更多的背景知识，另一方面，当你面临一个问题时，你可以很容易地找到相应的文档。

系统要求

.NET 框架可以安装在 Windows 98、Windows 98 第 2 版、Windows Me、Windows NT 4（所有版本）、Windows 2000(所有版本)，Windows XP(所有版本)以及 Windows .NET Server Family 服务器。

.NET 框架SDK 和Visual Studio .NET的系统要求是 Windows NT 4（所有版本）、Windows 2000(所有版本)，Windows XP(所有版本)以及 Windows .NET Server Family 服务器。

目 录

第一章 .NET 系统架构基本概念

第二章 .NET 多层架构应用开发

>> 逻辑层

提供了.NET与COM+服务集成中的详细技术信息，介绍了.NET Remoting框架的基本原理，主要组件及.NET Remoting与分布式对象通信的几种方案。

>> 表现层

介绍了一些重要的控件开发技术，指导你有效地开发自定义控件。

第三章 .NET应用程序的架构要点

>> 调试

Visual Studio .NET实现了三个高级调试目标：所有语言共用一个用户界面；跨机器同时调试多个进程的能力；强大的跨语言交叉调试。同时，附带的Exception Management Application Block提供了一个虽简单但是可以扩充的异常处理框架。

>> 性能以及性能监视

性能问题主要是为了解决由于软件的发展速度远远超过了硬件的发展速度所引发的瓶颈问题，它普遍存在于计算机系统架构／程序开发的方

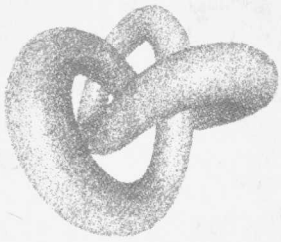
方面面。

>> 团队开发管理

针对.net 团队开发项目，首先需要理解在团队环境中如何建立开发过程，需要知道如何使用 Visual Studio.net 集成开发环境支持的团队开发性能(Visual SourceSafe)，同时还要意识到：你的开发团队成员必须遵守相同的开发规则，这样才能保证成功的团队开发工作。

>> .NET 应用部署：生命周期指南

该指南为成功地计划并部署.NET 应用程序提供了指导性原则、建议以及技术信息。



第一章

.NET 系统架构基本概念

>> .NET 系统架构基本概念

.NET的推出革新了软件开发和部署的很多观点，本系列文章首先带你理解这些概念。

服务(Services)
消息(Messags)
契约(Contracts)
策略(Policies)
状态(State)
进程(Processes)
应用(Applications)
术语表

导 读

简介

Web Services时代的来临给我们引入了一系列的新概念，比如Web Services、消息、状态以及过程等，本系列文章将对这些概念一一进行阐述。

服务

服务是应用逻辑的分散单元，它能把基于消息的、适合通过网络访问的接口展示出来。基于服务(service-based)的架构允许非常灵活的部署战略；这个服务模型允许应用来平衡网络计算资源，而不是要求所有的数据和逻辑驻存在一台计算机上。本文描述了一个基于互通性服务上的应用架构，它能够提供高效的业务逻辑和状态管理。

消息

这里所说的服务模型(service model)的含义是：服务由消息定义为其

能够接收和产生哪些消息的需求。服务间的成功的消息是个复杂的过程，它们是由跨服务的消息架构来处理。本文详细探讨这些概念，

契约

服务之间的通信是通过服务接口来进行的，服务接口能够发送和接收消息。服务到服务(service-to-service)通信遵循一个契约，并通过把这个契约公开化，它能够不必使交互行为做出让步而对服务实现作出变动。

策略

服务需要管理和进行安全配置。策略包含有一组规则，每一个规则都应用到运行时行为的某个方面。比如，一个服务可能有多个接口；你可以有能够应用到全部服务的规则，也可以有仅适用于一个或多个接口的规则。

状态

服务管理着状态，状态就是服务存在的原因。服务保护着状态，通过它们的业务逻辑确保状态保持连续和精确。状态是当前信息的唯一真实来源。

过程

业务过程控制着执行工作的逐步操作，把系统从一种状态转移到另外一种状态。在每一步，它都会调用一个业务操作。这些过程在一个业务过程服务或者过程服务上可以控制。在哪样的一个过程服务中的过程能够发送消息，以调用包含在服务中的业务操作，然后进入下一步，它可能需要使用不同的服务。

应用

在基于服务的架构里，应用是由过程服务以及更多的实现业务功能和用户接口的基本服务所组成。这种模型对传统应用（有和业务服务进行通信的用户接口）和业务到业务(business-to-business，业务服务和其他业务服务进行通信)场景都适用。

服务

服务（service）是现代企业应用程序架构的构筑模块。

软件服务是应用程序逻辑的非连续单元，它公开了适合通过网络访问的基于消息的接口。基于服务的架构允许非常灵活的部署策略，而不需要所有数据和逻辑都必须驻留在一台单独的计算机上，该服务模型允许应用程序充分利用网络化的计算资源。这种方法使得大量应用方案成为可能，下面是一些例子：

■ 多个客户系统之间共享状态。通过服务远程访问数据，许多客户端应用程序可以在共享的环境下工作，该应用方案适合于机构内部或者机构之间。

■ 将应用程序扩展到单个系统之外。服务使得集群和网格计算成为可能，从而能够支持极其复杂的应用程序，使得在单台计算机上操作大数据集成为可能。

■ 简化应用程序逻辑的部署。不需要在可能需要应用程序逻辑的所有计算机上都安装应用程序逻辑，应用程序逻辑可以在需要时简单地从远程访问得到。

■ 优化专用系统上的处理。数据存储和视频展现都是应用程序逻辑的极好例子，这些应用程序逻辑可以通过为专用目的而进行硬件设计和

优化来实现。

上述应用方案正在推动软件设计沿着将应用程序逻辑作为服务来建模和交付的方向发展。即使本地运行的软件也可以作为服务来设计，以便它们可以根据需要采用不同的配置来分发，或者根据随时间的变化的系统需求进行分发。

随着网络变得更快、更便宜和更可靠，服务架构也在不断演进。当前，主要有三种从服务抽象衍生而来的模式：

■ 远程过程调用(RPC)体系结构使用基于消息的服务来模拟本地方法调用。

■ 类似于RPC，分布式对象系统使用消息来模拟本地对象操作，包括方法调用和数据成员访问。

■ 以文档为中心的服务使用消息来执行受协议限制的会话，通常将命令和数据作为与平台无关的文本来传递。Internet基本的体系结构依靠这些服务来实现互操作性，具体的例子包括域名系统(Domain Name System，DNS)、使用简单邮件传输协议(Simple Mail Transfer Protocol，SMTP)实现的邮件系统、使用超文本传输协议(Hypertext Transfer Protocol，HTTP)实现的应用程序，以及可扩展标记语言(Extensible Markup Language，XML)Web服务。

虽然所有这些模式都将继续在应用程序架构中发挥重要作用，但是XML Web服务特别地提供了几个重要的优点。XML Web服务提供平台无关的方法来展现应用程序逻辑，从而带来更好的互操作性、功能需求与操作需求的分离以及独立的软件单元管理。通过明确地强调服务的wire format——状态管理的的XML表示，XML Web服务避免了平台相关性，这是制约其他分布式架构的互操作性的主要因素。

本文描述这样一种应用程序架构，它依赖可互操作的服务提供高价值的业务逻辑和状态管理。在不考虑所采用的特定技术的情况下，“服务”的概念应该包括下面这些特征：

■ 平台无关性。为了实现相异系统的集成，服务必须使用平台无关的数据表示，并支持访问应用程序逻辑的平台无关的方法。XML Schema(XML模式)规范就是用于数据表示的平台无关类型系统的一个例子。完全通过发送和接收的消息来定义服务，而不是将它模型化为一组功能调用，这样有助于实现访问应用程序逻辑的平台无关性。

■ 依据的标准。服务的互操作性依赖于标准。几个必需的标准的例子是有线协议、消息模式和安全证书。

■ 从元数据处理中分离服务文档。成功的服务传送依赖于丰富、健

壮的体系结构提供诸如可靠的消息传递、安全性管理、登录和审计等功能。这样的体系结构又依赖于所请求的操作（比如路由信息和响应的可缓存性）的相关元数据的访问。元数据必需是基于标准的，并从消息体携带的特定于应用程序的状态中提炼出来。简单对象访问协议(Simple Object Access Protocol，SOAP)通过将元数据放进消息的头部元素中并将应用程序数据放进消息体元素中来实现服务文档与元数据处理的分离。

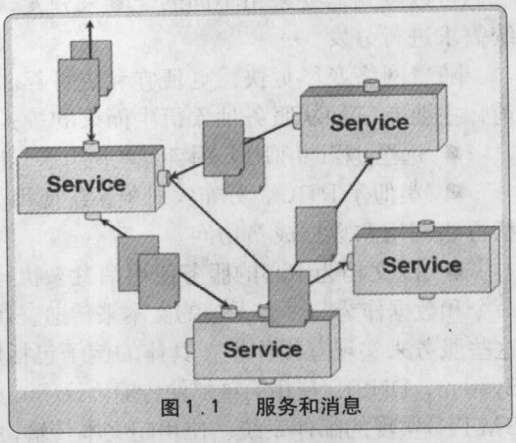


图1.1　服务和消息

在这些原则基础上构建服务，能够在其他类似的服务的动态网络中互操作，图1.1描述了基于对数据、元数据或两者的通常理解的消息路由。

服务模型

典型情况下，服务既提供业务逻辑，又提供与设计服务来解决的问题相关的状态管理。在设计服务时，目标是有效地封装逻辑和现实过程的相关数据，从而就服务应该包括的内容和应该实现单独的服务内容做出明智的决策。

由于服务被设计成通过网络调用，它们通常应该是粗粒度（Coorse-grained）的。也就是说，服务应该封装应用程序逻辑的基本主体，从而交付与网络请求的延迟成本相称的价值。类似地，服务应该公开粗粒度的接口：与公开许多接口（其中每个接口只操作少量数据）不同，服务应该公开较少的接口，允许单个请求执行一次完整的查询或更新。

因而，服务和传统组件之间的主要区别在于调用语义。对象通常是在调用时实例化的，并具有相对细粒度（fine-grained）的方法；而服务通常是长期运行的，它们要操作大量的消息。值得注意的是，服务调用并不意味着用实例化对象来处理消息，而是将消息作为标记（token）来处理，或者在进行完整的处理之前处理消息的特定片断（比如安全证书），这样做是合理的，也是常见的。

服务必须要非常注意保护它们的管理状态，对授予读写访问权限要极

其谨慎，并验证更新操作是否符合完整性规则。服务是它们所管理的状态的壁垒，对如何操作状态拥有绝对的权威。可以说服务对需要访问它们的外来者持一种“健康的怀疑态度”。

状态操作受业务规则（business rule）控制。业务规则是与状态关联的相对稳定的算法（比如根据项目列表汇总发票的方式），并且通常实现为应用程序逻辑。

服务受策略（Policy）控制。策略不如业务规则稳定，可能是地区性或特定于客户的。例如，某些优惠客户可能获得商品和服务折扣。一个非常另类的例子是管辖访问服务的SSL的使用策略。策略通常在运行时通过查找表来驱动(虽然需要应用程序逻辑查找并应用策略)。

因此，服务的更加完整的定义可能是：“服务是具有网络能力的软件单元，它实现逻辑，管理状态，通过消息通信，并受策略控制。”

这个定义所包含的每个概念将在后面部分深入讨论。“消息”部分将详细讲述消息和消息处理，“策略”部分将讨论策略，“状态”部分将探讨由服务管理的几种状态类型，“过程”部分将考察基于服务的架构中的分布式过程管理。

下面将描述与服务的设计和采用基于服务架构的技术好处相关的高级概念。

服务设计概念

服务是复杂的。优秀的服务设计要求软件架构师理解机构内部和跨行业部门的数据建模和业务过程。消息模式需要谨慎地塑造（factor），以复用公共元素并精确地表示与被建模的业务过程相关的状态和逻辑。跨部门和机构边缘工作的服务必须处理信任、身份验证和授权，在公共网络上运行的服务必须处理环境的不可靠性和可能存在的破坏活动。服务必须设计为能以最小的维护成本伸缩和可靠地运行。

提供关于如何处理这些复杂性的详细指导超出了本文的范围，不过介绍一下服务设计中的几个基本问题是有益的。本文集中的后续主题将进一步分析如何考虑服务设计。

功能需求和操作需求的分离

考虑应用程序的策略（包括开发管理及维护的成本），应该在操作体