希望

.NET开发专家

应用Web Services构建多层架构的高效.NET应用

--XMLChina论坛开发纪实

/HOPE/.NET/programming

更多图书请访问 www.bhp.com.cn go

HOPE programming

项目规划 | 软件架构 | Web服务 | 设计与实践 | 部署与应用

验证我们的工作请访问
www.xmlchina.net

北京希望电子出版社 总策划
王瑄 李燕 编著

科学出版社
www.sciencep.com

.NET开发专家

应用Web Services构建多层架构的高效.NET应用

--XMLChina论坛开发纪实

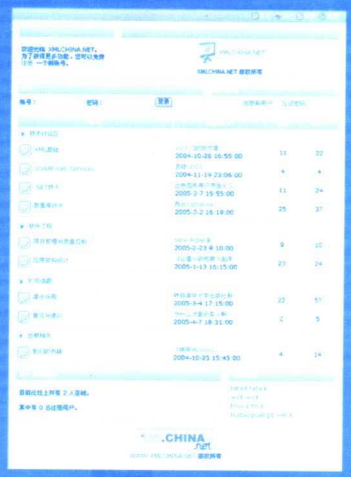
/HOPE/.NET/ programming

更多图书请访问 www.bhp.com.cn go

HOPE programming

项目规划 | 软件架构 | Web服务 | 设计与实践 | 部署与应用

以.NET项目开发为主线



验证我们的工作请访问
www.xmlchina.net

北京希望电子出版社 总策划
王瑄 李燕 编著

科学出版社
www.sciencep.com

内 容 简 介

本书讲述如何利用.NET、SOA 与 Web Services 构建企业级应用解决方案，本书第 1 部分介绍.NET 开发的相关背景和必要知识，从本书的第二部分开始，将以大型实际应用为例，按照分析和规划→设计和实现→部署与应用的思路展开讨论，这是讨论问题并迅速掌握技术的一种较好的方式。

本书只有一个示例（工程）——XMLChina 论坛。本书并不侧重于介绍某编程语言在软件开发中的技巧，而且本书也不试图向你提供可以被当作“字典”查询的编程方法。本书的侧重点是如何开发完整的、高效的企业级.NET 软件应用系统完整开发流程，为此，我们完成了 XMLChina 论坛这样一个完整的、高效的 Web 应用系统，请访问 www.XMLChina.net 验证我们的工作并参与讨论。

本书的大部分内容，是以 architect 和 analyst 的角度作为出发点，通过实际案例分析来展开的；在设计与实现部分，也包括了大量的软件系统开发介绍和 C#代码实现；可作为架构师、系统分析员、开发人员、或业务分析人员的阅读参考。

本书相关代码请到 www.b-xr.com 或 www.XMLChina.net 下载。

图书在版编目（CIP）数据

应用 Web Services 构建多层架构的高效.NET 应用—XMLChina 论坛开发纪实 / 王瑄　李燕 编著. —北京：科学出版社，2005.6

（.NET 开发专家）

ISBN 7-03-015299-9

Ⅰ.应...　Ⅱ.①王...　②李...　Ⅲ.互联网络-网络服务器　Ⅳ.TP368.5

中国版本图书馆 CIP 数据核字（2005）第 026898 号

责任编辑：大成　/ 责任校对：李兴旺
责任印刷：双青　/ 封面设计：刘孝琼

科学出版社 出版
北京东黄城根北街16号
邮政编码:100717
http://www.sciencep.com
双青印刷厂 印刷

科学出版社发行　各地新华书店经销

*

2005 年 6 月第　一　版　开本：899×1194
2005 年 6 月第一次印刷　印张：26 3/8
印数：1-3 000　字数：733 500

定价：46.00 元

与 潮 共 舞

—— 序言

什么是企业级应用架构?
什么是优秀的企业级应用架构?
.NET 下如何构架优秀的企业级应用架构?
SOA 是什么?
SOA 是对 OO 的否定么?
如何看待 SOA 与 Web Services?
何时以及如何应用 SOA 和 Web service?
……

在很多人心中，都陆续有过这几个问号（或者，也充斥着其他疑问）。也许你还在画第一个问号，也许你已经在思考最后的问号，甚或你早已对这些问题有了一种恍然大悟的感觉。那么，不管你对构建企业级应用解决方案有什么想法、对微软的.NET 战略有什么见解、对 SOA 和 Web services 了解到什么程度，都欢迎你阅读本书，与我们一起来探讨企业级 Web 应用及其应用架构。

无论采用什么技术平台，构建企业应用架构都不是一件容易的工作。回顾 IT 的发展历史，我们中的大部分人都经历过从两层式（client-server）到 3 层/n 层分布式架构的发展，IT 老手曾经玩转过 DOS 下的单机系统，对 FoxBASE 之类深有感触，资深前辈们也许还对打孔机记忆犹新……（打住！说的太多，有的读者恐怕要抱怨了："我们要学习怎样开车，不需要了解轮子是怎样发明的！"是的，就像 CPU 早已从 X86 进入 Pentium 时代，我们更应该做的，是紧跟发展的脚步，站在技术的前沿，与潮共舞）。言规正传，每当 IT 转型进入新的架构，都会有争议、探索，最初的跌跌撞撞恐怕是每个人、每个企业都会碰到的。

那么，面对.NET、SOA 和 Web Services 这些新技术的到来，我们需要做好什么准备？对构建应用架构来说，SOA 和 Web Services 能为我们带来些什么？这一次，我们在思维上和观念上将要经历什么转变？做出多少调整？

要想更好的了解 SOA 和 Web Services 架构，我们认为首先应该看清 IT 业的本质：IT 本来就是藉由信息科技的技术优势，来协助业务单位达到更高的营运效率及更好的服务质量，从而获得更大的投资利润和价值。打个简单的比方，如果 Internet 不能为人们带来种种好处，没有人能够从中获得使用价值，那么 WWW 也将不复存在。这应该是商业本质的回归，因为只有找对了出发点，我们才能正确的应用 IT 新技术。脱离了实际生活和业务运转的本质而单纯强调新技术的功用，就可能陷入象当年.com 神话那样的漩涡，成为快速破灭的泡沫（最终还是要回归价值）：皮之不存，毛将焉附？

本书内容

从本书的第一章开始，我们将对有关软件应用架构的问题逐一进行探讨和解释，期望与读者一起，将心中的每个问号都演变成句号、感叹号。

由于我们的讲述重点是如何利用.NET、SOA 与 Web Services 构建企业级应用解决方案，因此，一个应用实例自然是必不可少的：从本书的第二部分开始，我们将以实际应用为例，按照分析和规划→设计和实现→部署与应用的思路展开讨论，希望这是讨论问题的一种较好的方式。

本书的组织方式

理论来源于实践，它是对实践的提取，这决定了它的精简和不详尽，很多问题只有在实践中才能暴露

出来，从我们写作的角度说，很多东西光介绍理论是说不清楚的，所以理论必须联系实践。

很多书的作者（不是全部）显然和我一样认识到了实践的重要性，他们中的大多数完美地将软件开发的实例融入了枯燥的理论文字中，生动地向读者进行知识灌输，也有一部分作者很“实在”地将大段大段的代码象理论文字一样印刷在书中，如果你耐心地将他们印刷在纸上的代码全部输入计算机，最后你会惊喜地发现，这些代码居然编译通过了，而且还运行正常！

我可以很遗憾地告诉你，读我们的这本书，可没有这么便宜的事情！如果要将本书涉及的所有代码都印刷在书上，姑且不论能否通过出版社编辑部的“五关六将”，单单是要我夜以继日的 Ctrl-C 和 Ctrl-V，我也不干！所以，你在本书中能看到的代码仅仅是那些“关键”代码，你可以在 WWW.XMLCHINA.NET 找到完整的项目工程和代码。

如果你已经习惯了在随书光盘中找到按照章节划分的众多的代码示例（这种书很多，《XX 编程 X00 例》通常是它们的名字），那么你在观看本书时不得不对这种习惯做一些改变：本书只有一个示例（工程）——XMLChina 论坛。本书并不侧重于介绍某编程语言在软件开发中的技巧，事实上，在编写本书的示例代码时，我还要求开发小组尽量避免使用技巧而使用中规中矩的方法，而且本书也不试图向你提供可以被当作“字典”查询的编程方法。本书的侧重点是如何开发完整的、高效的企业级.NET 软件应用系统，为了让你感觉到这一侧重点的存在，我们“炮制”了 XMLChina 论坛这样一个完整的、高效的 Web 应用系统，本书从这里开始，就拿它“说事”了。

2001 年的时候，我做了 XMLChina 论坛的第一个版本，由于那个“实习”作品的存在，XMLChina 的成员们友好地满足了我的虚荣心，将本书中所涉及的系统命名为 XMLChina 论坛 2.0。在我们的设想中，XMLChina 论坛 2.0 没有华丽的外表和复杂的功能，因为这个版本的它最重要的作用是辅助我们在书中完成对软件开发的讲解；它应该结构清晰、采用先进的架构，因为它要足够经典，以对读者有示范意义；它应该代码工整、尽量使用中规中矩的方法实现，因为我们有义务让每一个读者都可以读懂它；它在功能上应该类似于“动网论坛”和“CSDN 论坛”的结合体，并且便于扩展，以满足 XMLChina 日常业务的需求；它应该有足够好的性能表现，满足大量用户的同时使用，因为也许有人想把它应用于大型的系统中。所以，我们要有良好的系统分析和设计。

幸运的是，几年来，众多规范而富有创意的.NET 系统开发经历使 XMLChina 的成员们个个都精于此道，与他们在一起工作，简直就是一种享受。

读者对象

本书的大部分内容，是以 architect 和 analyst 的角度作为出发点，通过实际案例分析来展开的；在设计与实现部分，也包括了大量的软件系统开发介绍和 C#代码实现；可作为架构师、系统分析员、开发人员、或业务分析人员的阅读参考。

如果你是一名初学者或入门者，可以阅读第一部分，相信能够从中获得有用的知识。但在阅读其他内容之前，你应该具备适当的.NET 编程基础、关系型数据库知识、以及软件设计基础。

特别说明

不管是哪一个技术阵营，或者某种技术方向，都应该有相应的权威人士和经典著作，如果你希望深入的学习某个技术架构或是某种开发语言，请参考那些方面的经典著作（比如《Applied Microsoft .NET Framework Programming》、《Thinking in Java》、《Microsoft SQL Server 7.0 Programming Unleashed》、《Systems Analysis and Design in a Changing World》等等）。在这里，我们恐怕无力超越这些权威或经典，我们所做的仅仅是作为 IT 行业中的从业者，从使用和实用的角度出发，把一些经验和认识看法拿出来与大家共享，共同交流，如果读者能从中得到一些启发和收获，在软件开发过程和应用 SOA 与 Web Services 的过程中少走弯路，将是我们得到的最好回报；如果读者有更为深刻的看法和建议，请不吝赐教、参与讨论，那是我们莫大的荣幸；如果发现错误和勘误，请向我们指出，这将是对我们最大的支持和帮助。

如果有需要交流的地方，欢迎发 email 到 xmlchinanet@hotmail.com 或访问我们的网站 www.xmlchina.net

在本书中，为了便于读者理解，对于可能产生歧义或有多种称法的词汇，在其第一次出现的地方，我们一律标出了英文原词；一些关键词汇，在多数地方均采用中英文并陈的方式。

特别声明：在本书内容中涉及到的人名、机构名称、产品、角色或数据等信息纯属虚构，如有雷同，实属巧合，除非另作说明，否则他们不代表任何真实的人物、机构、产品或事件，并没有任何故意的“影射”或歧义。

感谢

感谢 XML_China 团队的所有成员，特别感谢年轻而优秀的程序员韦超，他是一位值得信赖和依靠的合作伙伴，他为本书所做的贡献不仅包括不停被采纳的意见，还有为实用的 Web 表现层和传说中的 XML_China 论坛助手编码。

感谢希望出版社，感谢编辑栾大成先生，没有他的认真和宽容的态度，本书无法完成；同时还要感谢在编写本书过程中给予我们帮助的同事和朋友：孙涛、梅雪峰、倪凯、金殷勇、徐晓峰、周雄立等。

程序员推荐序

从基本的面向对象（OO），到基于构件（Component）体系架构的提出，人们已经在信息系统构架和开发方法上前进了一大步。在短短的几年内，三大技术体系在分布对象技术和构件化服务方面都取得了不凡的成就，它们分别是 OMG（对象管理集团）的 CORBA/CCM（公共对象请求代理体系结构及其构件模型）、SUN 倡导的 J2EE/EJB 架构、微软公司的 COM/COM+/.Net 体系结构。应该说，这三大技术体系都有自己的特点和优越性，到底孰优孰劣，存在很大的争议；经过几年的争战，它们也都在现今的 IT 市场占有了一席之地。一般来说，微软.Net 架构在中小企业规模应用上占有优势，而 J2EE/EJB 架构在大型信息系统建设中更受青睐，至于 CORBA，虽然已没有了前几年的那种风光，但仍在电信、金融等领域有着重要的应用。

“存在即为合理”。为什么人们一定要争个你死我活呢？有什么方式可以和平共处吗？令人欣慰的是，面向服务的体系架构（SOA）为实现技术融合提供了希望。

SOA 架构及其主要实现途径——Web 服务（Web Services），可以让各个企业和软件研发公司选择最适合自己的技术架构和开发方法来建立企业内部的信息系统，同时又能够方便地获取专业公司提供的 IT 服务、实现和客户以及交易伙伴的无缝信息交互。它就像一种粘合剂，把原来斗得头破血流、非此即彼的技术体系（比如 J2EE 和.Net）通过一种合作的方式拉在了一起，共同为建立数字化社会添砖加瓦。使用 SOA 和 Web 服务，不仅企业内部可以实现不同类型信息系统的有机集成，甚至在不同企业之间也可以实现信息的互通和互操作。

微软.Net 架构是实现 Web 服务及 SOA 的重要技术途径。本书的主要内容，即是讲述如何利用微软.Net、SOA 与 Web 服务来构建信息系统，提供企业级应用解决方案。书中对有关软件应用架构的问题逐一进行了探讨和解释，并以一个实际应用为例，按照分析和规划、设计和实现、部署与应用的思路进行了深入的讨论。

我对这本书另一个深刻的印象，则是它的原创性。目前，IT 业的兴旺和图书市场的繁荣，已经让十几年前为了寻找一本技术专著而搜遍大小书店、甚至不惜求助国际友人从国外获取的现象一去不复返了。然而，图书种类的爆炸性增长，又给读者增添了新的烦恼：同一技术主题往往存在着几十甚至几百本不同类型的图书（译著、专著、编著、原版），想从浩瀚书海中挑出适合自己的那一本，实在是大海捞针、难上加难。

国内绝大多数的软件技术类图书都是舶来品，特别是近年来，几乎国外各知名出版社的每一部软件图书一经面世，就很快在国内有了引进中文版。遗憾的是，为了抢占先机、占领市场，国内很多出版社只顾抢速度，却忽视了译者的水平和翻译质量，造成大部分的译著都好似“天书”一般令人头痛，甚至根本就把原著的关键描述翻译错误。于是，英文水平好的技术高手宁可去读原汁原味的英文版本，而大部分英文不佳却又对软件技术情有独钟的读者只好硬下头皮去逐字逐句的“猜测”式学习了。

与此同时，在为数不多的国内编著书籍中，又存在着太多的“剪刀＋浆糊”的写作手法，大量重复国外著作的内容和知识，缺乏自己的真知灼见和实践经验。随着国内 IT 业的飞速发展，越来越多的程序员正在成熟起来，他们在应用开发实践中逐渐积累了大量经验和教训。可惜的是，许多这样的经验知识往往散布于各个论坛、BBS 甚至技术高手自己的“脑子”中，很少见到专门的技术书籍来总结这些宝贵的实践经验。

本书正是弥补这种不足的一个努力。作者在介绍 SOA 及 Web 服务的技术概念和开发方法的同时，把自己在 XMLChina 论坛项目中的开发经验有机地融入到全书中，试图以实践的方式更有效地阐述如何使用微软.Net 架构设计和实现 Web 服务的技术、方法和途径。

我以为，软件开发是一种实践性很强的技术，离开实践，空洞地介绍技术体系往往是苍白无力的。尽管本书作者在对技术的理解和把握上可能还存在不足甚至是错误，但他们确确实实是把技术实际应用在具体的项目中，并从中获取了经验和教训。因此，读者更可能从这本书中获得共鸣、得到帮助，找到有价值的闪光点。

希望作者付出的努力能够获得回报，也希望看到更多由国内程序员原创的书籍不断涌现。

李霖

个人简介

李霖，男，1972 年出生，33 岁，1998 年 11 月毕业于国防科技大学计算机学院，获博士学位，副研究员，在北京系统工程研究所从事软件技术总体工作，研究兴趣包括软件架构、中间件、SOA 及 Web 服务、软件测试及质量保证。在《计算机学报》等学术刊物和会议上发表论文 20 余篇，出版学术专著 2 部（合著），获得部委级科技进步一等奖 1 项、二等奖 2 项、三等奖 1 项。

值得一提的是，作者常年战斗在技术最前沿，是国内最早接触.NET 框架核心的几位程序员之一（其他几位都是微软中国公司亚洲研究院人员），是国内.NET 技术阵营公认的大师级程序员。

目　录

第一部分　基础

第二部分　设计与实现

第三部分 部署与应用

第四部分 深入话题

第一部分 基 础

第 1 章 软件应用架构

“我们用什么架构、用什么技术平台来实现这个应用？”当你面对用户的需求，开始准备为一个企业应用设计解决方案时，首先呈现在脑海里的，也许就是这个问题。

应用架构是软件系统的核心支柱，就像骨架对于人体有着不可替代的重要性：架构的好坏往往从一开始就决定了整套系统解决方案的成败。

架构又依赖于技术，二者是相辅相成的。随着计算机的诞生，软件系统也开始发展；随着硬件处理能力的不断飞跃，软件系统的复杂性也不断增长，为了提高软件质量、缩短软件的开发周期、更好的满足企业应用需求，各种开发技术、开发方法也在不断的创新、发展和完善。技术和方法的发展，不断提高了软件的使用性和生命力。

往往是需求推进了发展，这里，我们又一次强调了应用本质的回归，IT 技术同其它行业技术一样，最主要的目的和用途是“服务”——服务于商业领域、生活领域、科学探索……在生活中的方方面面、在任何角落，几乎都可以享受到 IT 技术带给我们的种种好处。IT，可以当之无愧的称之为第三次工业革命。IT 加快了世界前进的脚步，软件改善了商业和企业的运营模式。当然，软件的应用架构也需要满足企业的需求、符合企业的运行结构，成功的软件不应该只是时髦的概念、整齐的文档、值得炫耀的漂亮编码，最最重要的，是满足不断发展的用户和企业的需求。否则，软件就可能成为“一笔失败的投资”。

IT 技术的发展日新月异，它的变革速度，可能是所有行业中最快的，今日的 IT 巨人，很可能就是明日的一撮黄土。然而，真正突破性的技术变革并不多，大部分技术都是不断的改良改良再改良：结构重新调整一下、性能再优化一些、实用性和易用性再加强一些、等等。软件及其架构，也是在变化中求得发展，从而更快更好的为企业应用服务。

1.1 软件体系结构的发展

谈软件的发展，也许会让你觉得有点多余，觉得是老调重谈、没有必要。然而凡事有因才有果，知其然才能知其所以然：只有真正了解了软件设计发展之路，你才能正确理解 Web Services 和 SOA，从而把握趋势，正确的应用 Web 服务架构，而不是为了 Web Services 而 Web Services。

软件行业有着与生俱来的特殊性，技术发展快，从业人员流动性大，用户的需求变化也快（当你花费 1 年的时光完成一件软件产品时，很可能用户的需求早已经大相径庭）。现在，行业的竞争又越来越激烈。因此，如何提高软件质量、积累知识财富、提高可复用性、提升开发效率、尽可能缩短开发周期等课题，成为软件企业成败的关键因素。

从起点到达目标的最佳路线无疑是直线，“两点之间直线距离最短”——这是我们在中学时代就已熟知的真理。为了简捷而快速的达到预期目标，IT 从业人员也在不断的总结经验、应用一些好的模式和方法。于是，各种开发技术、开发方法、应用架构应运而生，这些设计和实现的最佳实践通常会被表示成软件的设计模式。

这一节，我们就来简单回顾一下软件设计和应用架构的发展历程。

1.1.1　软件设计的发展

从出现第一台计算机以来，计算机时代已经经历了半个世纪的发展和变化，特别是从 Dos 时代开始，软件便成为一匹技术行业的黑马，迅速成长和发展；软件的应用，已渗入到各行各业，对我们的生活和工作方式产生了巨大影响。

在软件设计的初期，用户总是和一个单块的程序交互，那时一个程序往往就代表了一个软件，这个独立的程序通常包括所有的代码：应用逻辑、数据处理、界面、通信等。这个时期是软件行业的个人英雄主义时代，一个项目或程序，总是只由一两个编程高手独立完成（实际上，这种情况即使在现在也不难看到）。

麻烦的是，所有功能代码往往都混杂在了一起，就好像一团乱麻，或者说，就像把许多外形相似、但疗效不同的药丸放在一个盒子里。你可以想象，要修改或解决一个问题将是多么困难。

结构化设计

随着计算机硬件的突破性发展和性能的飞跃，操作系统和软件的功能也不断发展和完善，它们能做的事情越来越多，应用领域不断扩大，人们对软件的期望值也越来越高，期望把更多更难更复杂的事情交给软件去解决，并希望藉此提升效率、创造更多应用价值。这样，软件的规模和复杂度便与日俱增，甚至超出了当时业界的技术和方法所能驾驭的程度。

经过多年的总结，大家发现，典型的软件结构，几乎总是多多少少的包含那么几个部分：应用逻辑、数据管理、通信、用户交互界面。

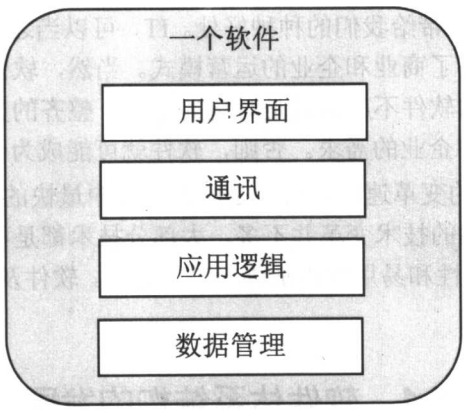


图 1-1　典型的软件结构

当通过函数的结构化封装揭示出一个程序的固有逻辑，人们就会发现，一个小游戏的通信方法和一个航空线路计算程序的通信方法几乎是相同的，甚至是使用相同的通信协议；于是问题被提了出来：有必要在每个程序中都重新编写功能差不多相同的函数吗？

结构化程序设计，是软件设计领域迈出的重要一步。结构化设计的核心思想，可以概括为：

程序 = 算法 + 数据结构

这种设计思想的中心是对数据流的分析，通常也被称为是面向过程的分析设计。数流图和数据字典成为结构化设计文档中的主要部分。

面向对象的分析设计

随着社会和软件的不断发展，人们不断感觉到结构化设计也不能很好的表达事物、描述需求，需要从更高的层次上进行思考，从比函数更高的级别上进行封装和描述。这时人们对系统建模的切入点逐渐从数据流向对象（Object）演变。打个比方，我们应该都听过瞎子摸象这则寓言，假设我们正面对一头大象，如果采用结构化分析思想，我们可能会向摸象的瞎子一样，忽视了大象的完整个体表现，首先去关注血液是如何流动并发挥作用的；而采用面向对象的分析思想，我们首先关注的，则是大象这个动物本身——一

个完整的对象。

面向对象产生的直接原因是为了提高程序的抽象程度，控制软件的复杂度。公认的面向对象建模语言出现于 70 年代中期。90 年代后期，UML 建模语言的形成，逐渐成为工业界的标准。UML，相信大家都不陌生，它是现在被广泛使用的面向对象建模语言。

面向对象（Object Oriented）的建模方法发展至今日，可以说颇为成熟，利用对象的思想为软件系统建模，已经成为软件开发的主要工作，而传统的编码工作却“退居二线”了。一个系统的模型建立的好，就为满足用户需求、保证系统的稳定性和质量、提高系统的扩展性打下了良好的基础。

面向对象思想的核心，也可以简单概括为几个术语：对象、类、封装、继承、多态性。

对象是面向对象开发方法的其本元素，是一个封装了数据和操作这些数据的逻辑实体，它是运行期的基本实体。每个对象都可以用它本身的一系列属性和其上的一系列操作来定义。对象可以是现实生活中的一个物理对象，也可以是某一类概念实体的实例（Instance）。比如，一辆汽车，一个人，一台电脑，乃至一个图形、一个界面都可以作为一个对象。

类是对一组具有相同属性和操作的对象的抽象，是对对象共同特征的描述。比如：电脑可以被定义为一个类，具体的某一台电脑就是一个对象。

在一个类中，每个对象都是类的实例，可以使用类中提供的方法。要想从类的定义中产生对象，必须有创建实例的操作。

下面我们举一些例子来简单说明一下封装、继承和多态性：

- 多态性

  多态是指一个事物具有不同表现形式的能力。类的多态性体现在类可以实例化成多个不同表现的对象，下图就是一个简单的例子：

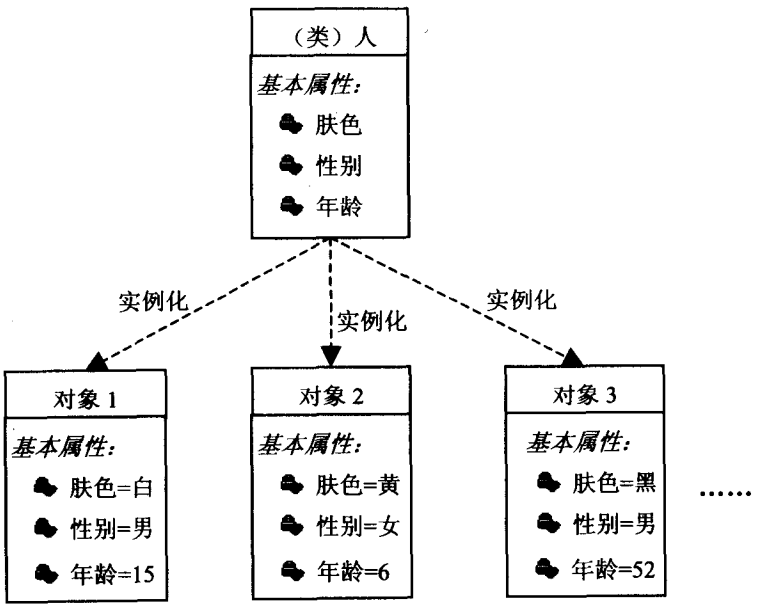


图 1-2　类的多态性

- 封装性

  一个类封装了一组属性和针对这些属性的一组操作。封装将数据和代码捆绑到一起，使其成为相对独立的实体，避免了外界的干扰和不确定性，从而提高聚合性、降低耦合性。操作可以理解为一些方法和函数。很明显，封装的层次已经远远超越了单个函数的概念。

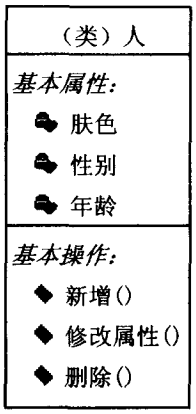


图 1-3 一个完整的封装类

- 可继承性

  继承在遗传学里很容易理解：我们每个人都会从父母那里继承一些基因特点，比如外貌、性格、口音等等。在软件设计中，继承也是相似的概念：继承是让某个类型的对象获得另一个类型的对象的全部特征，继承者将获得被继承类的所有属性和方法，同时，它还可以拥有自己的新特性。

面向对象提高了对系统的封装性，使软件复用可以从函数级别跨越到类的级别，与结构化设计比较，面向对象的设计更易于实现对现实世界的描述，使软件系统的结构更加清晰而利于理解，也更容易满足实际应用的需求。因而得到了迅速发展，对整个软件开发过程产生了深刻影响。

面向服务的分析设计

面向服务（Service Oriented）是最近提出的理念，它是一种更高层次上的封装。从微软的最新视角来看，服务是中心，我很认同这个观点。要讨论这个问题，我们必须首先回归到基本的出发点：网络技术和信息技术为什么而存在？从商业模式的角度看，因为有价值而存在，有价值可以理解为：提供某种为人所需的服务。服务我想应该不需要再解释了，每个人都曾经使用过别人提供的服务，也能随口说出几个服务的例子，比如，google：好使的搜索服务；楼下的烧饼铺子：好吃的早餐服务；EMS：国际化的快递服务。

面向服务架构（SOA，Service-Oriented Architecture）强调透过服务的概念来提供 IT 的各项基本应用功能。有人认为 SOA 是对 OO（面向对象）的颠覆和否定，但我们不这么认为。实际上，SOA 与 OO 并不矛盾，都是分析问题、架构系统的方法，只不过，SOA 是另外一种角度的封装。一个服务可能需要多个对象来协同实现，一个服务也有可能会包含（调用）其它服务。

谈到 SOA，不可避免的要提及 EAI（企业应用集成），软件业发展到现在，造就的专有名词，如 ERP、CRM、SCM、CMS……，多到目不暇给。在大家试图为不同的企业应用提供解决方案的同时，也累积了一个个的信息壁垒，不同系统、或者新旧系统之间的信息通讯成了极大的困难。SOA 的重要目标，就是以服务为导向在软件系统之间搭建横向的沟通桥梁。

SOA 又总是经常与 Web 服务（Web Services）一起被提及，Web 服务可以说是扮演了 SOA 催化剂的角色。那么，如何看待 SOA 与 Web 服务呢？在本书的第二章中，我们将着重讨论 SOA 与 Web 服务，以及 Web 服务的核心标准。

1.1.2 软件应用架构的发展

伴随着软件设计方法的发展，应用架构的变化也经历了类似的历程。最初，人们使用的都是哑终端、单机系统，不需要考虑什么应用架构的问题：简单的任务往往只需要由一个独立的程序来实现。然而，当软件能作的事情越来越多、软件的复杂程度越来越高时，应用架构就成了必须要考虑的事情。这就好像盖房子，当原始社会的人们对房子的要求只是遮风挡雨时，盖房子也只是简单的工作：一两个人随便搭个棚

子就可以。而当代的人们对房子有了不同的要求，我们需要有公寓楼、别墅、办公大厦、购物中心、大剧院……，这时，设计一份良好的蓝图就成了关键的必要工作。

主机/终端结构

自上个世纪六、七十年代开始，主机/终端架构逐渐形成，这些人们用来与计算机交互的哑终端是非智能的。这种应用架构以典型的批处理、联机交易、消息传递和数据库为技术方向，也被称之为单层模式系统。企业也纷纷采用这种架构构建自己的应用。

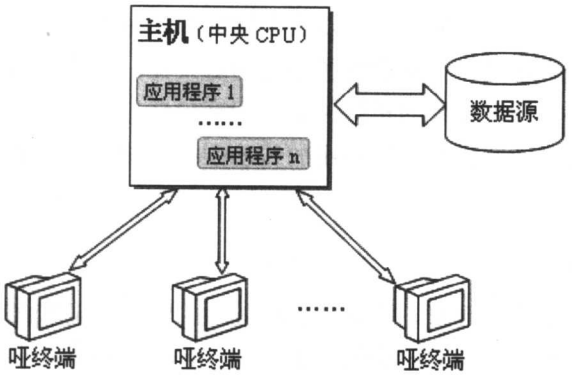


图 1-4　典型的主机/终端模式系统

客户端/服务器结构

随着个人计算机（PC）和局域网的出现，信息处理资源开始由远程计算机（服务器）和本地个人计算机（客户端）共享，在服务器和客户端运行的软件之间需要建立通信链路。这就是典型的 C/S（Client/Server，客户端/服务器）架构。

在 C/S 模式下，服务器通常采用高性能的 PC、工作站或小型机，并采用大型的数据库系统。客户端需要安装专门的客户端软件。这种应用架构以前台图形界面（GUI）和后台关系型数据库为技术方向。C/S 模式大大降低了企业构建业务系统的门槛，使得企业可以采用性价比更好的方式去构建自己的部门级和企业级业务应用系统。

C/S 架构的优点是可以充分发挥客户端的处理能力，很多应用业务逻辑可以在客户端处理后再提交给服务器，这样客户端的响应速度通常也比较快。

C/S 模式的缺点主要是：客户端需要安装专用的客户端软件，只适用于局域网环境。系统的规模越大，则系统的灵活性就越差，管理越麻烦。客户端/服务器结构的可扩展性、可维护性较差，且数据通讯和交换能力不够，远程数据访问需要专门的技术，要对系统进行专门的设计来处理分布式数据。

浏览器/服务器结构

随着互联网的飞速发展，Web 浏览器（Browser）迅速成为人们桌面上不可或缺的工具，移动办公和分布式应用越来越普及。这时，新的远程访问方式、新的应用模式 B/S（Brower/Server）开始大行其道，迅速成长起来，而互联网的廉价和无处不在则更是推动了这一切变化和发展。

在 B/S 模式下，客户只要只要安装一个 Web 浏览器（比如 Internet Explorer、Netscape Navigator 或 Opera 等等）来下载 HTML 页面，而所有的应用业务逻辑和数据处理都放在服务器端。浏览器通过 Web 服务器（Web Server）与服务器进行交互。

B/S 模式的最大优点就是不用安装任何专门的客户端软件，只要有一台能上网的电脑就能使用，客户端实现零维护，从而系统的扩展也非常容易。