"九五"国家重点电子出版物规划项目
计算机知识普及系列

2001 编程宝典 丛书 2

自由编程世界奇葩

E-Commerce Application System with CORBA

用CORBA创建电子商务系统

北京希望电脑公司 总策划
刘 晖 沈钧毅 林 欣 编 写

本光盘内容包括
1 本版电子书
2 本版书中的实例程序源代码

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

“九五”国家重点电子出版物规划项目

计 算 机 知 识 普 及 系 列

2001 编程宝典 丛书 2

自由编程世界奇葩

E-Commerce Application System with CORBA

用CORBA创建电子商务系统

北京希望电脑公司 总策划

刘 晖 沈钧毅 林 欣 编 写

本光盘内容包括

1. 本版电子书

2. 本版书中的实例程序源代码

北京希望电子出版社

Beijing Hope Electronic Press

www.bhp.com.cn

内 容 简 介

本盘配套书是专门介绍用 CORBA 创建电子商务系统的书，内容包括在 Intemet 上采用 CORBA 开发分布式系统的技术内幕；从程序员角度介绍了电子商务的有关内容，提出了电子商务软件的基本要求，同时详尽介绍了 CORBA 在 C++ Builder 中的实现。

全书由 11 章组成，第 1、2 章是 CORBA 体系及其采用的技术，并对 CORBA 的接口定义语言 OMG IDL，接口存根（IDL Stub）及其接口框架（IDL Skeleton）进行了介绍；第 3、4、5 章对 OMG IDL 在标准 C 及 C++中的映射技术，CORBA 中的 ORB 客户端、服务器端进行了详细阐述；第 6、第 7 章是 CORBA 互操作技术，包括各种层次上的桥接、GIOP/IIOP 等协议，并提出了电子商务简介及其对分布式软件的基本需求；第 8 章系统描述了 CORBA 的各种基本服务；第 9、10 两章结合大量实例介绍了 C++ Builder 的 CORBA 编程技术，包括改进后的 ATM 电子商务演示系统和拥有强大数据库功能、“主从”关系表以及公文包模型的网上售书电子商务演示系统。本书最后一章对 CORBA 与现有分布式软件开发技术进行了比较，并介绍了 CORBA 的新技术。

本书内容新、丰富，结构清晰、图文并茂，并附有大量实例，实用性、操作性和指导性强，对读者整体把握 CORBA 编程的核心思想提供了很好的帮助。本书既可以作为从事电子商务系统开发、编程的广大从业人员实用的自学指导书，也可以作为高等院校相关专业师生教学、自学用参考用书和社会相关领域培训班教材。

本光盘含本版电子书及配套书中实例程序源代码。

系 列 盘 书：	2001 编程宝典丛书 2
盘 书 名：	用 CORBA 创建电子商务系统
文本著作者：	刘晖　沈钧毅　林欣　编写
文 本 译 者：	希望图书创作室
CD 制 作 者：	希望多媒体创作中心
CD 测 试 者：	希望多媒体测试部
责 任 编 辑：	纪红
出版、发行者：	北京希望电子出版社
地 址：	北京中关村大街 26 号　100080
	网址：www.bhp.com.cn　E-mail: lwm@hope.com.cn
	电话：010-62562329,62541992,62637101,62637102,62633308,62633309
	（发行和技术支持）
	010-62613322-215（门市）　010-62521798（编辑部）
经 销：	各地新华书店、软件连锁店
排 版：	希望图书输出中心
CD 生 产 者：	北京中新联光盘有限责任公司
文本印刷者：	北京媛明印刷厂
规 格 / 开 本：	787×1092　1/16 开本　16 印张　373 千字
版 次 / 印 次：	2000 年 12 月第 1 版　2000 年 12 月第 1 次印刷
印 数：	0001～5000 册
本 版 号：	ISBN 7-900056-26-2/TP・26
定 价：	30.00 元(1CD，含配套书)

前　言

这本书的主题就是电子商务系统中的 CORBA。我们将分别介绍 CORBA 的来历、CORBA 的详细模型、CORBA 与 COM/DCOM 的对比。在提供足够的信息用来把握隐蔽在各种编程语言、开发工具、体系结构之后的“关键线索”的同时，也会介绍 CORBA 在 C++Builder 中的实现。本书包括以下内容：

- OMG 及其倡议 CORBA 的动机、CORBA 采用的技术、CORBA 体系概述。
- CORBA 的接口定义语言 OMG IDL、接口存根 (IDL Stub) 与接口框架 (IDL Skeleton)。
- OMG IDL 在标准 C、C++中的映射。
- CORBA 中 ORB 的客户端，包括静态调用接口 (SII)、动态调用接口 (DII)、接口仓库 (Interface Repository)、异步通信方式。
- CORBA 中 ORB 的服务器端，包括对象适配器 (BOA)、实现仓库 (Implementation Repository)、服务器启动策略、动态框架接口 (DSI)。
- CORBA 互操作，包括 CORBA 中 ORB 与 ORB 的调度、各种层次上的桥接、GIOP/IIOP 等互操作协议。
- 电子商务简介及其对分布式软件提出的需求。
- CORBA 基本服务，包括对象生存期 (Lifecycle) 服务、对象关系 (Relationship) 服务、持续对象 (Persistent Object) 服务、对象外化 (Externalization) 服务、对象命名 (Naming) 服务、对象洽谈 (Trader) 服务、事件 (Event) 服务、事务 (Transaction) 服务、并行 (Concurrency) 服务、属性 (Property) 服务、对象查询 (Query) 服务、对象包容 (Collection) 服务、对象安全 (Security) 服务、对象时间 (Time) 服务及许可 (Licensing) 服务。
- 用 C++Builder 实现 CORBA，包括 VisiBroker 的 Smart Agent、接口仓库、Object Activation Daemon、各种程序向导以及静态、动态程序实例。
- 两个完整的电子商务演示系统实例，包括改进后的 ATM 电子商务演示系统以及拥有强大数据库功能、“主从”关系表以及公文包模型的网上售书电子商务演示系统。
- CORBA 与现有分布式软件开发技术的对比，包括 CORBA 与 COM/DCOM、CORBA 与 Java、CORBA 与 Web；同时介绍了 CORBA3.0 的一些新动向。

由于 CORBA 是与 COM/DCOM 相互竞争、相互补充的技术，它们之间的比较也出现在本书的不同部分。

接触过 CORBA 的朋友都曾经为之激动振奋过。CORBA 不是一种编程语言，而是一套在 Internet 上实现分布式系统开发的思路、方法和工具集。CORBA 本身仍然在高速发展着，而促进它发展的技术原因就是所有编程语言、开发工具、体系结构都在碰到、都未能（也不可能）单独解决的问题。

CORBA 既是计算机软件行业过去十年发展历程的总结与见证，也是计算机软件行业解决现有技术问题的新方法新思路。但愿 CORBA 成为我们驰骋 Internet 分布式编程、纵横电子商务软件开发的利器。

2000 年 8 月 28 日

目　录

第 1 章　Internet 需要 CORBA

本章内容提要

- 对象管理组织——*OMG*
- 为什么需要 *CORBA*
- *CORBA* 依赖的技术
- *CORBA* 概述

1.1　对象管理组织 OMG 提出了 CORBA

1990 年 11 月，对象管理组织（Object Management Group）在《对象管理体系指南》一书中首次提出“公共对象请求代理结构”（Common Object Request Broker Architecture），以后被人们简称为 CORBA。

对象管理组织 OMG 的 800 多名成员遍布世界各地，其中 5%来自亚洲、澳大利亚、非洲及南美洲，30%来自欧洲，其余来自美国本土。他们包括 DEC、Microsoft、Netscape、Oracle、Novell、IBM、Inprise、Informix、Iona、Hewlett-Packed、Rogue wave、Sybase 、Sun 等许多著名公司。

不过，OMG 并不开发软件，仅仅制定指标。OMG 成员每年聚会 6 次，提出自己对各种软件标准的建议。这些提议在经过具有“协作资格”成员充分地讨论、修改后，由 OMG 董事会公布为 OMG 官方标准，并由标准倡导者在一年内推出标准的相关实现产品。任何公司决定采用 OMG 标准时，既不需要 OMG 的认可，也不需要交纳任何费用。

CORBA 标准的提出，是软件界的一场革命。

1.2　CORBA 的用途

Internet 使计算机联结起来，CORBA 则使应用软件联结起来。没有操作系统的计算机几乎没有用处，没有应用软件相互集成的 Internet 也很难显示出互联网的终极魅力。在 Internet 上集成应用软件，就是通常所说的“分布式软件开发”。CORBA 正是为满足这种源自 Internet 的需要，实现软件全方位集成而设计的。

分布式软件开发需要解决一系列的兼容问题，包括以下五个跨越：

- 跨平台。未来的软件将分布在各种机型的平台上：大型机、PC 机、笔记本、带程序的电视机、录像机、传感器、报警器、各种 DSP、PDA 等等。
- 跨操作系统。计算机世界的操作系统种类繁多，有称霸 PC 机的 Windows 系列、源远流长的 Unix 及其变种、蓄势待发的 Linux、经典的 Solaris OS 等，分布式软件开发必须正视这一现实问题。
- 跨语言。至今为止，还没有一种通用、万能的计算机编程语言。Java 可谓灵巧，但目前还是不适合于开发用户界面；Delphi、C++Builder 后来居上，却缺乏 Java 跨平台、跨操作系统的兼容性；Visual Basic 虽然开辟了可视化编程的先例，却无法胜任大型软件的开发；Visual C++则神情严肃、过于呆板苛刻，缺乏 C++Builder 的平易近人和灵活便利……未来的世界，允许混合编程，使每种语言都在最擅长的领域一显身手。
- 跨协议。Internet 是一个异质结构的网络，在不同的区域可能具有不同的网络结构、传输协议，为了使软件运行时具有数据、方法共享性，操作透明性，集成软件时

必须考虑协议不同带来的不便。

- 跨版本。用户对软件功能的需求总是在逐步增加，每次变化都会要求软件工程师重新编写程序模块。分布式软件开发会给软件的版本更新带来不便——应该让多少客户进行 Update？又会导致多少软件模块发生连锁性重写？因此，在 Internet 上集成软件必须实现版本的透明性。

到目前为止，CORBA 是用来解决以上五个跨越问题的唯一方案。

作为分布式软件的用户，CORBA 可以使我们得到以下好处：

- 我们可以使用各种信息、数据，无论它们分布在数字世界的哪个角落。
- 我们可以对信息、数据进行各种自动化处理，无论它们需要哪个公司的哪个软件。这种处理是无缝集成的，就好象使用专门编制的软件一样。
- 我们可以随意更改事务处理流程，实行物业动态管理。这种改变、革新对办公室自动化、电子商务的现有设施影响很小。我们可以最大限度的重用已经存在的软件、硬件和数据信息。
- 我们可以使用、控制各种数字化的设施：嵌入程序的智能电视、录像机、报警系统、传感器以及各种几乎类似“职业选手”的“个人数字助理”——PDA 等只要这些带程序的嵌入式数字化设备接入了互联网。

作为分布式软件的开发人员，CORBA 可以使我们获益如下：

- 混合编程。我们可能偏好某种编程语言，嗜好某种开发工具，但我们需要与具有不同癖好的伙伴合作开发大型软件。
- 丰富的编程元素。任何符合 CORBA 的软件模块一旦问世，就会成为可以被重用的资源。无论何时，无论何地。
- 高效率的开发手段。使用 CORBA 方式编程，不但软件模块的重用十分便利，而且软件模块天生具有无缝集成性。第二个 CORBA 软件开发人员就不再是“高楼万丈平地起”，软件开发只需要解决还没有被别人解决过的问题，编写别人还没有编写过的代码。
- 版本无关性。使用 CORBA 方式编程，版本不但具有向下兼容性，而且具有向上兼容性。也就是说，Version2.0 的用户自然能够获得 Version1.0 的功能，同时，Version2.0 的用户也可以调用未来 Version3.0 的新功能。

实际上，CORBA 将在 Internet 上实现软件的即插即用。

1.3 CORBA 采用的技术

CORBA 首先是一种编程技术，是吸收了软件界面向对象技术、分布式计算技术、多层体系结构技术以及接口技术的一种综合技术。

1.3.1 CORBA 采用了面向对象技术

CORBA、Java、COM/DCOM 都采用了面向对象技术，而且它们都已经或正在将面向对象的概念上升到比“类”更加高的一个级别——“组件”（Component）。

面向对象技术涉及许多抽象的定义，对此我们不想过多讨论。但是，面向对象技术是 CORBA 存在、实现的形式，因此简要说明如下：

- 类。类是一个程序集合，它记录了被表达概念或实体的共同特性及共同操作、处理过程、函数。前者形成类的数据，后者成为类的方法。比如，表示客户的类至少包括“客户名称”这一共同特性，称为类的数据。而告诉别人自己的名字以及改变自己名字的函数、过程被认为是“客户类”的方法。
- 对象。对象是类的具体实例，给类中的数据变量赋予确定的取值便得到该类的一个对象。比如，“张老三”、“王老五”就是“客户类”实例化后产生的两个不同对

象。编写程序时，考虑的是类；运行程序时，处理的是对象。不过，在许多场合，人们并不刻意区分类和对象这两个概念。

- 封装。封装是实现类的机制，确保了不同类之间的相对独立性。比如，通过封装，“客户类”和“商品类”相互独立。此时，“客户类”在未授权的情况下不能直接使用“商品类”的数据；“商品类”的修改（增加、更改类中的数据或方法）也不会直接影响“客户类”。
- 继承。继承反映了不同类之间的一种派生关系。比如，电视机首先是一种商品，具有商品的所有特性，其次又含有指定的尺寸、体积、接收装置；遥控电视机也具有电视机的全部特性，同时还增加了遥控设备。通过继承，可以实现代码重用。与以往的子程序模块相比，继承保证了代码重用时的逻辑关系，提高了可移植性和安全性——通过确定继承关系，目前正在编写的代码将与能够重用的代码自动联系起来。类之间还存在多重继承关系。比如，“电视机类”可以从“商品类”和“无线电设备类”共同派生出来，表示电视机是一种作为商品出售的无线电设备。
- 重载。重载体现了类内部及派生类之间的差别，包括函数重载、操作符重载。函数重载可以使类中同名方法具有不同操作。比如，商品类与电视机类中都可以有一个称为 Show 的方法，前者可以在屏幕上显示商品的名称，而后者则在屏幕上直接绘出电视机的外形。客户类还可以有两个带有不同参数的 Name 方法，当参数为计算机屏幕时，在屏幕上显示自己的名字；当无参数时，通过多媒体读出自己的名字。另外，加减乘除等运算符号均可以借助操作符重载在各个类中实现不同的操作。
- 多态。多态反映了类中方法在实际调用过程中呈现的随机性和不确定性。如果发现商品是电视机，我们可以附带赠送一年的电视报；如果发现商品是 DVD，赠送物品则变为一盒影碟不过无论是哪种商品，都需要付款。这时，可以采用多态技术，在运行过程中动态地选择具体操作方法。多态包括虚函数、重置、虚继承等情况。

以上是面向对象技术的一些主要概念。

注意 CORBA2.0 是完全基于面向对象技术的。目前，CORBA3.0 正在朝着基于 Component 的方向发展。Component 可以认为是更高级别上的“类”。不过，真正的类往往用同一种语言实现继承、重载、多态；Component 则可以在不同语言中实现。另外，Component 具有属性、事件。实际上，Component 是经过包装处理的一组类，阅读完本书就会理解这一点。COM/DCOM 是基于 Component 的一种分布式编程技术。本书将不断对比 CORBA 与 COM/DCOM 的差别。

1.3.2 CORBA 采用了分布式计算模型

CORBA、COM/DCOM 都采用了分布式计算模型。IBM 曾经认为，计算机世界是一个“英雄创造历史”的世界，他们仅仅重视大型计算机——处理和控制都高度集中的主机。IBM 的失误造就了 Microsoft 和 Intel 两个巨人。现在，计算机世界都垂青于分布式计算模型——资源、功能、任务、控制等统统分布的系统。

分布式计算模型具有以下特点：

- 分布性。数据、处理、控制并不驻留在某个站点，而是比较均匀的分布在整个互联网上。因此，功能、任务也因而分布开来。
- 并行性。不同任务可以并行执行。既可以在同一站点并行执行，也可以通过数据、计算迁移实现多站点并行执行。
- 透明性。分布式计算隐藏许多了内部实现细节，包括物理位置、并发控制、错误处理。这些隐藏实现的是应用性能上的透明性。

- 共享性。软件、硬件的各种资源高度共享。
- 强健性。数据、处理和控制的分布，使系统故障得以分散。一般情况下，局部故障不会给整个系统造成太大影响。同时，整个系统的整体容错性、可靠性也相应加强。

实际上，CORBA从一开始就是为分布式软件开发、集成而设计的。

1.3.3 CORBA采用了多层体系结构

CORBA、COM/DCOM都采用了多层体系结构。

电子商务软件一般可以划分为三大模块：用户界面、商务规则、信息数据管理。而电子商务软件的开发却已经经历了单层体系结构、双层体系结构、客户/服务器体系结构、三层体系结构、多层体系结构几个阶段。目前，在如何划分多层体系结构上还存在分歧。这些体系结构之间的差异和分歧在于如何定位、调度、集成三大模块。

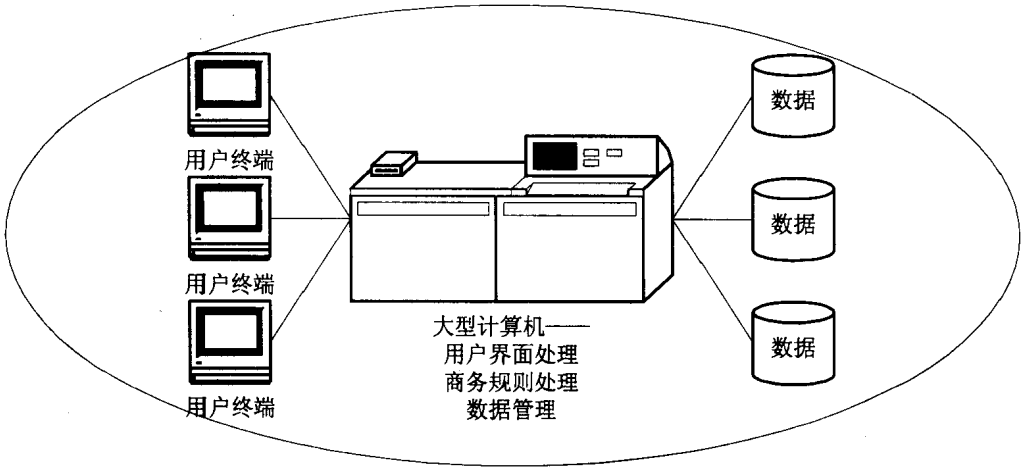

图1.1 单层体系结构

图1.1表示单层体系结构。用户终端仅仅用来显示、接收操作信息，大型计算机将三大模块处理集于一身，数据被存放到指定位置。虽然大型计算机终日忙忙碌碌，可用户还是觉得它行动迟缓；而且，任何用来完善用户界面的代码都会被认为是一种奢侈和浪费。

随着PC机的发展，人们将用户界面处理部分的代码移入了用户终端，形成了双层体系结构。后来，PC机性能更加卓越，人们甚至将一部分商务规则处理代码也移入了用户终端，形成了客户/服务器体系结构，如图1.2所示。

客户/服务器体系结构在实际应用中不断受到两个事实的挑战：

- 电子商务规则日益复杂。软件开发人员不可能或者很难再将部分商务规则移入小型工作站、高性能PC机。
- 电子商务规则经常变化。软件开发人员需要不断更新电子商务软件算法，而且应该及时完成。

为了有效解决上述问题，人们决定采用三层体系结构，主要包括下列三层：

- 用户界面层，仅处理图形用户界面。
- 商务规则层，仅处理商务规则。
- 数据管理层，仅实现数据管理。

与此同时，面向对象技术、组件（Component）技术相应诞生，类、组件成为人们进行软件开发的主要手段，成为构造三层体系结构软件的核心元素。

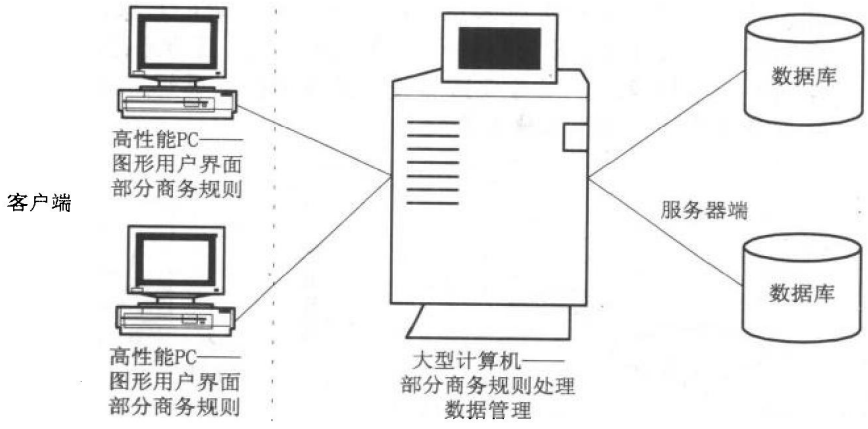

图 1.2 客户/服务器体系结构

接着，人们发现需要位于不同机器上的类、组件能够相互通信，这就导致了中间层——商务规则层的继续分化。它们从一台机器上移到了 Internet 上，相应产生了多层体系结构。也有人认为，多层体系结构就是三层体系结构。这种分歧并不重要，总之电子商务软件要“分布”进行。

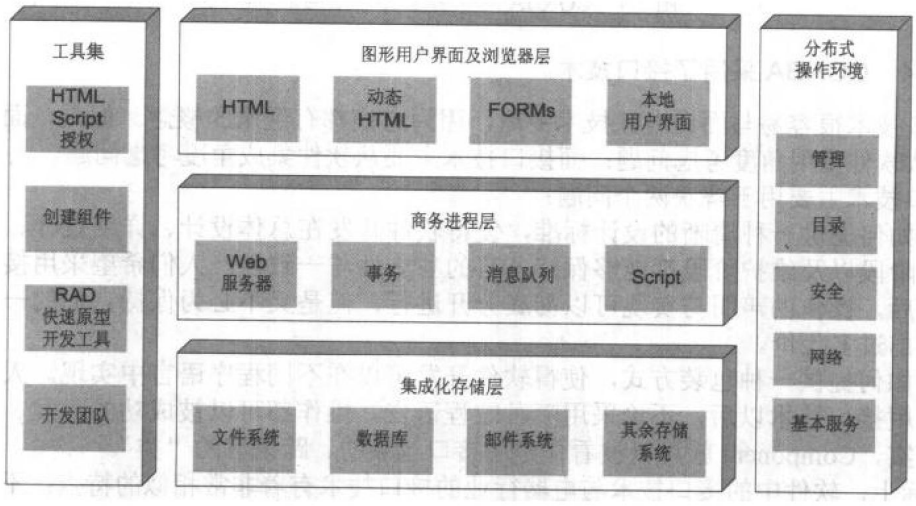

图 1.3 Microsoft 对于多层体系结构的理解

Microsoft 认为三层体系结构就是多层体系结构，包括客户层、组件层和数据管理层。它们配合 Microsoft 的各种产品，形成由 COM/DCOM 统一支配的分布式软件开发结构，如图 1.3 所示（就是现在流行的 ASP）。

CORBA 认为的三层或多层体系结构如图 1.4 所示。而 CORBA 本身是位于中间层的中间件（Middleware），是关于分布式对象、分布式组件的标准。

注意 CORBA 与 Microsoft 对于多层体系结构的理解在后两层产生了分歧。Microsoft 的第三层只有数据，其它服务都归入第二层；CORBA 的中间层更加关心的是对象的互操作性，因此，资源层依旧包括与数据有关的服务。这种分歧与支持这两种模型的软件公司所采取的经营策略密切相关。Microsoft 希望所有的客户都使用自家的产品，因此在第二层里囊括了全部服务，由 Microsoft Transaction Server 实现。而 CORBA 是公司联盟产物，希望各种服务由不同的公司提供。因此，在中间层更加关注产品的兼容、通用性，并认为商务规则可以通过对各种资源的有效集成来实

现。这些资源，可以来自地球上任何一个有智慧的厂家。作者认为，从面向对象技术角度出发，CORBA似乎更为合理，因为无论是类、组件都符合这样一条规律：数据与数据有关的操作不能从逻辑上分离。也就是说，数据与它们的服务很难被划分到不同层。

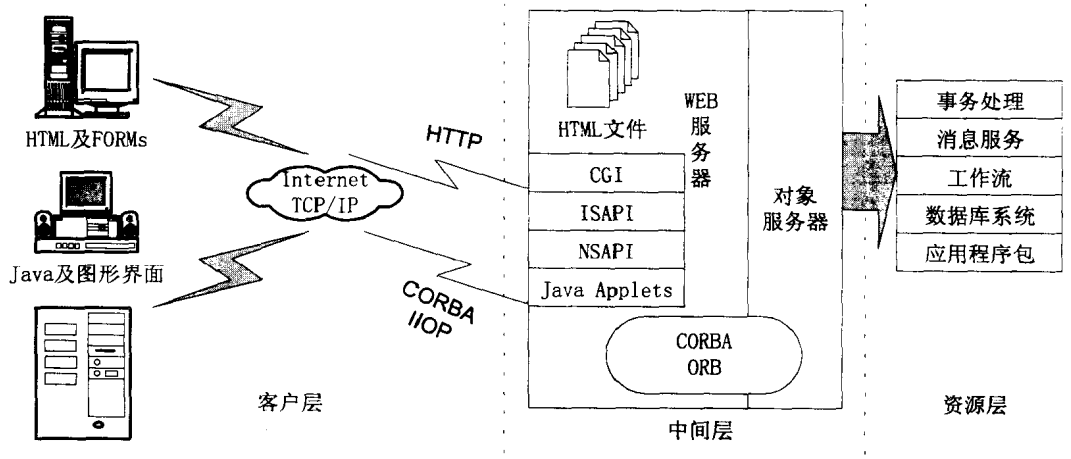

图1.4 CORBA对于多层体系结构的理解

1.3.4 CORBA采用了接口技术

接口技术很容易与面向对象技术混淆，因为它们都有继承的概念。但是，面向对象技术主要从软件重用角度考虑问题；而接口技术主要从软件集成角度考虑问题。

接口技术主要用于解决两个问题：

- 如何提供一种清晰的设计标准，使得软件开发在总体设计、详细设计、具体编码阶段以及维护阶段都能够保持各自的独立性与一致性。人们希望采用接口技术以后，操作的声明与实现可以彻底分开进行，但是又不必为保持它们的一致性而付出过多代价。
- 如何提供一种包装方式，使得软件开发可以在不同程序语言中实现。人们希望采用接口技术以后，不论采用哪种编程语言，操作都可以被成功的激发、调用。其实，Component就可以被看作是用接口包装的、跨语言的“类”。

实际上，软件中的接口技术与电器行业的接口技术有着非常相似的特点：不管厂家或软件开发人员是谁，都必需达到、完成接口中规定的指标及内容；不管厂家或软件开发人员采用哪种技术，从同一种接口中都应该获得相同的功能。

如果我们把接口当作一种规定和指标，就不难理解接口的以下一些特点：

- 接口本身不能包含数据变量。数据对应着状态，接口本身没有状态，因此也不能含有数据声明。但是，操作中有参数，因此，接口中也包括参数说明。
- 接口本身没有实现自己的能力。接口中所有操作、功能的实现都是通过其它软件模块、其它类来完成的。接口仅仅是一个“交流”的渠道，可以吞吐但无法生产。不过，接口可以强迫软件模块、类在编码实现中符合自身的规定。

1.4 CORBA概述

CORBA的最终目的是分布式软件集成。CORBA既代表了一种软件开发模式、一种软件开发标准，也提供了软件开发必需的服务、可以使用的工具集合。因此，CORBA整体上是由“对象请求代理”、“CORBA服务”、“CORBA工具集”与符合CORBA标准的

各种应用程序、对象一同综合形成的。如图 1.5 所示。

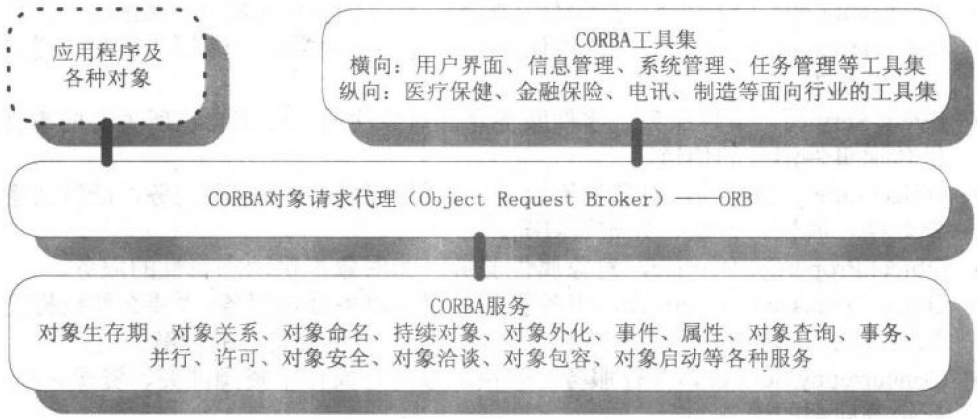

图 1.5 CORBA 的完整体系结构

其中，开发人员仅仅需要完成由虚线框起来的部分，其余均由 CORBA 本身提供。

1.4.1 CORBA 对象请求代理——ORB

在 CORBA 中，对象请求代理（Object Request Broker）被简称为 ORB。ORB 是联结应用程序、各种对象、CORBA 服务、CORBA 工具集的核心。当然，也正是 ORB 将这些构成 CORBA 图景的元素有序地分割开来，而这种分割是 CORBA 实现分布式软件集成和即插即用的全部秘诀。ORB 主要实现了以下分割（联系）：

- 对象方法、服务的“定义”与它们的“实现”之间的分割。通过接口定义语言 OMG IDL，我们可以获得规范、通用的对象方法、服务定义；借助 ORB，这些定义可以在任何编程语言、代码模块中真正实现。这种分割有助于进行具体软件编码互换、编程语言互换以及版本互换。
- 请求“客户”与响应“服务器”之间的分割。客户对其它对象方法、服务的请求并不直接传递给被请求服务器，而是转交给 ORB，由 ORB 监察服务器的位置，状态，决定服务绑定的方式。这种分割有助于对分布式对象进行跨平台、跨协议的“逻辑集成”，是一种“伪客户/服务器”方式。
- CORBA“基础设施开发者”与 CORBA“收益者”的分割。采用 CORBA 进行分布式软件开发的确是一个令人心醉的建议，但是，真正完全实现 CORBA 的所有标准难度很大。作为一般 CORBA 用户，我们都可以借助某种 ORB 产品来简化开发任务。ORB 就是某些软件公司为我们建造好的，进行 CORBA 软件开发的基础设施。

简而言之，ORB 可以帮助 CORBA 对象互相“理解”对方，可以在 CORBA 对象之间传递信息、请求，可以“管理”CORBA 对象之间的分布与集成。

1.4.2 CORBA 基本服务

CORBA 基本服务为基于 CORBA 的分布式软件开发提供了“对象”级别的服务。在 CORBA 中，有 16 个基本服务。

这些基本服务的英文、中文名称及简要说明分别如下：

- Lifecycle Service，对象生存期服务。提供与对象创建、移动、拷贝、释放等操作有关的服务。
- Relationship Service，对象关系服务。提供表达对象之间相互关系的服务。相信软件开发人员都明白，在“对象”世界中“乱搞关系”绝对会引起轩然大波，导致

系统混乱。

- Naming Service，对象命名服务。为对象提供姓名、别名、引用的服务。
- Persistent Object Service，持续对象服务。提供保持对象状态（数据）的服务。
- Externalization Service，对象外化服务。提供一种类似“流”的机制，提取、恢复对象的状态。
- Event Service，事件服务。事件服务允许对象注册、注销自己所关心的事件，并提供传递事件消息的信道。
- Object Query Service，对象查询服务。提供一种操作语言和服务，能够随意查询对象数据、属性、方法、分布等情况。
- Object Properties Service，对象属性服务。为对象提供动态属性的服务。
- Object Transaction Service，事务服务。提供事务处理服务：“要么统统提交，要么统统滚回”。这是电子商务对于“一手交钱，一手交货”的理解。
- Concurrency Service，并行服务。解决对象并行操作中资源冲突、资源共享、同步、死锁等问题的服务。
- Licensing Service，许可服务。许可服务允许用户只有在开发者认可的情况下才能使用对象，获取服务。
- Trader Service，对象洽谈服务。如果你知道你需要的服务，但不知道哪个是最适合的服务器、哪个是最经济的对象，就可以使用对象的洽谈服务。
- Security Service，对象安全服务。提供安全可靠的对象服务。
- Secure Time Service，对象时间服务。时间服务允许对象获取当前时间、确定事件发生顺序、计算时间间隔、实现基于时间的各种业务。
- Object Collection Service，对象包容服务。为对象提供数组、队列、栈、集合等容器。
- Object Startup Service，对象启动服务。决定对象如何启动以及启动时的状态。

以上服务是软件集成时，各种对象可能需要的公共服务。

1.4.3 CORBA工具集

CORBA工具集就是CORBA Facilities。它们为基于CORBA的分布式软件开发提供了“应用”级别的服务。比如，在办公室自动化领域，我们需要功能强大的文字处理功能、图形处理功能。软件开发人员可以从CORBA工具集中选择一些相关的工具，在此基础上开发应用软件，从而忽略繁杂的文字、图形底层处理细节。

CORBA工具集没有统一的标准，它们多少可以调用CORBA基本服务实现。但是，这些工具集合使基于CORBA的分布式软件开发不必总是“从头开始”，不必考虑一些过于原始、基本的问题。实际上，CORBA工具集是CORBA程序库。

CORBA工具集被分为横向工具集（horizontal facility）和纵向工具集（vertical facility）。CORBA横向工具集包括以下四部分内容：

- 用户界面（User Interface Common Facilities）。主要包括将数据转换为显示器、打印机等媒体支持的格式，管理图形用户界面等任务。
- 信息管理（Information Management Common Facilities）。主要包括建立各种信息模型、存储及恢复信息数据、支持多种信息模型的转换、支持不同数据格式互换、数据加解密等任务。
- 系统管理（Systems Management Common Facilities）。涉及内存管理、线程及进程池（Thread and process pooling）等能够降低分布式软件运行开销的任务。可以进一步分为服务质量管理（Quality of Service Management）、测量（Instrumentation）、数据集（Data Collection）、对象集合管理（Collection Management & Instance Management）、规划管理（Scheduling Management）以及系统安全管理（Security

Management）等内容。

- 任务管理（Task Management Common Facilities）。涉及工作流（Workflow）、代理（Agents）、规则管理（Rule Management）、对象自动化等任务。

CORBA 横向工具集涉及每个完整程序都需要考虑的问题：不论你正在开发什么软件，都需要在程序中解决一些类似的问题。比如，我们的应用程序都需要有友好的交互界面，需要通过多种方式显示同一份数据，需要驱动不同介质的存储器、光驱，需要管理特定的工作流、商务代理、商务规则以及实现对象自动化（类似 OLE Automation）等等。这些问题都可以从 CORBA 横向工具集中找到合适的解决方案。

CORBA 纵向工具集主要包括以下五个领域：

- 医疗保健（Healthcare）
- 金融服务（Financial Service）
- 电讯（Telecommunications）
- 电子商务（Business Objects）
- 制造（Manufacturing）。

CORBA 纵向工具集为工业界人士使用基于 CORBA 的软件提供了方便：在每个领域里，都有许多功能强大的应用软件。用户的业务既可以直接采用其中的工具，也可以在它们的基础上有所更新。每个领域中都没有特定的标准，只有可以方便地进行集成的各种服务和保证服务质量的各种许诺。

CORBA 工具集的目的十分清楚：当我们需要开发分布式软件时，可以从横向、纵向工具集中选择便捷的 CORBA 软件“部件”，快速地进行软件无缝集成。而且，任何功能强大、运行便利的 CORBA 对象都可以成为 CORBA 工具集中的侯选对象，甚至是 CORBA 工具集中的名牌对象。

1.5 本章小结

一个真正意义上的跨国组织——OMG 建议采用 CORBA 进行分布式软件开发。

因为 CORBA 可以最终解决跨平台、跨操作系统、跨语言、跨协议、跨版本等兼容问题，彻底地实现分布式软件集成。

CORBA 依赖于面向对象技术、分布式计算技术、多层体系结构技术和接口技术。

对于软件开发人员而言，CORBA 是一种分布式软件的开发方式，一种分布式软件的开发标准，也是一系列分布式对象公用的服务和一系列分布式软件开发的工具。

实际上，如果希望真正理解 CORBA 的奥妙，应该记住：CORBA 的精髓不在于如何开发软件，而在于为何能够这样开发软件。

第 2 章　CORBA 接口及接口定义语言 OMG IDL

本章内容提要

- *CORBA* 的伪“客户/服务器”方式
- *CORBA* 中的接口
- 接口存根（*IDL Stub*）
- 接口框架（*IDL Skeleton*）
- *CORBA* 接口定义语言 *OMG IDL* 扼要
- *CORBA* 伪对象
- *CORBA* 接口定义语言与 *Microsoft* 接口定义语言比较

2.1　CORBA 灵活的伪客户/服务器方式归功于 IDL

对象请求代理 ORB 是联结 CORBA 对象、CORBA 应用程序、CORBA 基本服务以及 CORBA 工具集的核心。在以 CORBA 方式开发分布式软件时，上述各个环节并不直接交互或者集成。ORB 将它们有序地分割，但又有机地联系起来。

如果从面向对象分析角度来考虑问题，CORBA 可以简化为如下表述：可集成的分布式对象（程序）和一个能够从逻辑上集成这些对象，被称做 ORB 的对象群，通过请求、响应的方式实现功能互联。如图 2.1 所示。

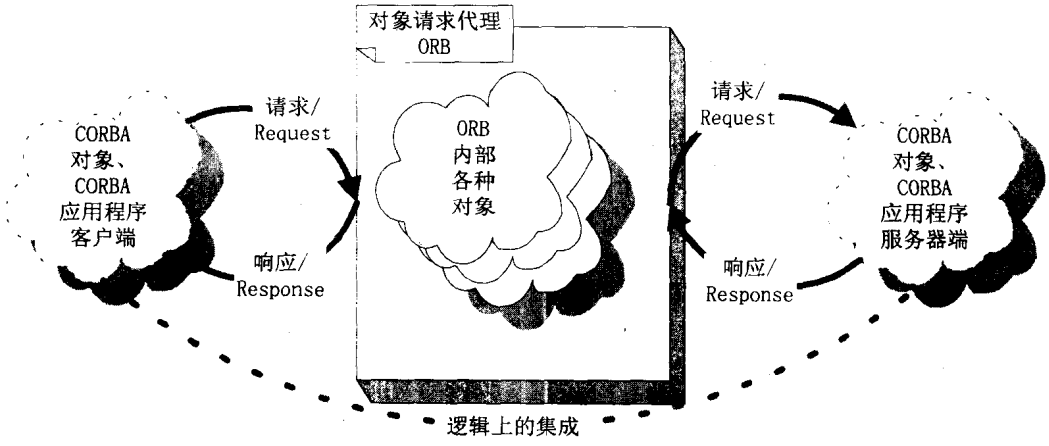

图 2.1　CORBA 的简化表述

分布式对象、程序之间的请求/响应方式就是传统的客户/服务器方式。但是，CORBA 中的请求“客户”和响应“服务器”之间并没有直接进行 “通信”。CORBA 客户向 ORB 发送请求，并从 ORB 接收响应；CORBA 服务器接收来自 ORB 的请求、调度，并向 ORB 发送响应。实际上，这是一种以 ORB 为中间件的伪客户/服务器方式。

CORBA 的这种伪客户/服务器方式具有更大的灵活性，可以满足分布式软件开发的下列现实需求：

- 允许分布式对象、程序具有灵活的角色。一般说来，客户总是提出执行某个任务的请求，服务器则进行相应的处理并发回有关响应。但是，一个对象、程序在现实中可能既是客户又是服务器。比如，全局数据库管理对象被动地接收、转储来

自局部数据库对象的更新数据时，明显是服务器；当全局数据库管理对象主动要求局部数据库对象提交更新数据并加以保存时，又可以看作是客户机。这种角色的歧义性在软件开发中经常发生。在 CORBA 中，只有客户——Client 与“对象实现”——Object Implementation，没有专门指明的服务器对象，就是为了消除这种歧义。

- 本地服务请求与远程服务请求的透明性。客户机与对象实现可以驻留在同一台机器上，可以分布在同一局域网内，也可以分散在互联网上。显然，针对具体情况，每次服务请求的具体操作可能不同。通过伪客户/服务器方式，这种不同由 ORB 监控处理。对客户而言，永远只有一种服务请求方式：CORBA 服务请求。
- 允许客户机与服务器之间保持灵活的对应关系。通过伪客户/服务器方式，客户机与服务器之间不需要保持一一对应。当客户工作量较大时，可以请求多个服务器同时提供服务（比如，文件下载时，可以申请多个“对象实现”同时进行）；当“对象实现”吞吐量较大时，可以并行服务于多个客户。虽然 ORB 不能自动实现负载均衡，却为它提供了足够的支持。
- 允许客户机与服务器在运行时构造新型的、动态的服务请求。伪客户/服务器方式不要求客户机/服务器之间直接进行一问一答的请求和响应，因此也不需要在程序编写时绑定客户机/服务器，可以在程序运行时动态提出服务请求。
- 允许服务器以多种形式存在。服务器可以是包含在单个或者多个进程中的对象、程序；可以是需要向别的服务器提出请求的中介对象、程序；甚至可以是一段需要激发的程序代码。ORB 可以获得必要的信息，并采取不同手段激发有关的对象、服务。
- 允许客户/服务器之间以同步、异步方式通信。一问一答的请求/响应方式大多以同步通信方式进行。由于 ORB 的中介，CORBA 伪客户/服务器方式却能够进行异步通信。包括所谓的延迟同步（deferred synchronous）和单向请求（one-way request）。

在我们感慨 ORB 带来如此奇妙的伪客户/服务器方式的时候，应该明白，这些功能的实现，首先是因为 CORBA 客户与 CORBA“对象实现”采用了接口技术，并使用统一的接口定义语言——OMG IDL 来描述各自的数据、结构和行为。本章以下部分就将着重讨论 CORBA 的接口和 CORBA 的接口定义语言。

2.2 CORBA 中的接口

按照面向对象分析的方法，CORBA 中至少应该存在三组对象（或者说“类”）：CORBA 客户对象、ORB 对象、CORBA 对象实现（或者说“CORBA 服务器对象”）。它们的关系如图 2.1 所示。

根据第一章提到的 CORBA 的五个“跨越”目标，我们很容易对这三组对象提出以下要求：

- 我们希望选择不同厂家提供的，或者不同语言编写的 CORBA 客户对象
- 我们希望选择不同厂家提供的对象请求代理 ORB
- 我们希望选择不同厂家提供的，或者不同语言编写的 CORBA 对象实现

为了满足这三个要求，实现 CORBA 所期望的软件即插即用，我们发现，需要在以上三组对象之间引入新的对象，处理由于用户选择不同产品而导致的集成障碍，如图 2.2 所示。