分布式构件技术综述

韦华颖[1,2]，詹剑锋[2]，王 沁[1]

(1. 北京科技大学 信息工程学院 计算机系，北京 100083；2. 中国科学院 计算技术研究所 国家智能中心，北京 100080)

摘 要：分布式构件技术的出现和发展为网络计算平台上软件的开发提供了强有力的解决方案。讨论了构件的概念与特性，分析和评价了几个主流的分布式构件技术及其它们的应用现状，并探讨了与之相关的正在发展的一些技术。最后对新的软件开发方法：基于构件的开发的思想和发展前景进行了简单的分析。

关键词：分布式构件；EJB；CORBA/CORBA 构件模型；COM/DCOM；.NET；OGSA；基于构件的开发

中图法分类号：TP311.52 **文献标识码**：A **文章编号**：1001-3695(2004)10-0012-04

Overview of Distributed Component Technologies

WEI Hua-ying[1,2], ZHAN Jian-feng[2], WANG Qin[1]

(1. Dept. of Computer Science, School of Information Engineering, Beijing University of Science & Technology, Beijing 100083, China; 2. National Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China)

Abstract: The occurrence of the distributed component technologies provides a powerful solution for the software development across a network. This paper discusses the conception and characteristic of the component, analyses and evaluates some kinds of existing crucial distributed component technologies. At the same time, it presents the overview of the developing relevant technologies. At last an outlook for Component-Based Development is provided.

Key words: Distributed Component; EJB; CORBA/CCM; COM/DCOM; .NET; OGSA; Component-Based Development

分布式构件技术随着网络和面向对象技术的发展而不断地成熟。面向对象技术对工业软件开发过程有着重要影响，但是面向对象编程在软件生产效率和可重用性上不能达到满意的效果，由此触发了构件(Component)技术。构件技术通过封装一定的功能来提供第三方构件，构件本身不需编程改动就能采用或重用，对外提供服务。构件通常比传统的对象具有更大的粒度，这种特性使得构件能够复合多个对象的功能，对外提供服务。与传统的计算服务模式不同，分布式构件之间的关系是对称的，不同构件之间可以位置透明、语言独立和平台独立地互相发送消息、请求和提供服务。

1 构件的定义和特性

著名的 Brown 和 Wallnau 将构件定义为一个非平凡的、几乎独立的、可替换的系统组成部分，它在定义完善的体系结构环境中实现某一清晰的功能。OMG(Object Management Group)的定义更通俗、详细一些：构件是一个物理的、可替换的系统组成部分，它封装了实现体并且提供了对一组接口的实现方法。构件表示了系统实现体的一个物理片段，包括软件代码(源代码、二进制代码或可执行代码)或者等同体，如脚本或命令文件。构件自身必须相容于接口且实现接口，接口表示了驻留在构件内的成分所实现的服务，构件实例为客户端构件提供这些服务。总之，构件应该具有以下几个基本属性或特性：

(1) 构件是可独立配置的单元，因此构件必须自包容。

(2) 构件强调与环境和其他构件的分离，构件的实现是严格封装的，外界没机会或没必要知道构件内部的实现细节。

(3) 构件可以在适当的环境中被复合使用，因此构件需要提供清楚的接口规范，可以与环境交互。

(4) 构件不应当是持久的，即构件没有个体特有的属性，理解为构件不应当与自身副本区别，在任何环境中，最多仅有特定构件的一份副本。

2 主流构件技术分析

2.1 Enterprise JavaBeans

2.1.1 J2EE 平台

J2EE 是 SUN 公司 1999 年推出的一种全新的 Java 平台，用来解决企业级服务端计算的需要，J2EE 平台提供了基于构件的方法来设计、开发、装配及部署企业应用程序。J2EE 平台提供了多层的分布式的应用模型、构件重用、一致化的安全模型以及灵活的事务控制。

2.1.2 EJB(Enterprise JavaBeans)

EJB 是 J2EE 平台的核心，也是 J2EE 得到业界广泛关注和支持的主要原因。EJB 的核心思想是将商业逻辑与底层的系统逻辑分开，使开发者只需关心商业逻辑，而由 EJB 容器实现目录服务、事务处理、持久性、安全性等底层系统功能。

EJB 是服务器端的构件模型，它是一种特殊的、非可视化的 JavaBeans，运行在服务器上。EJB 同普通的 JavaBeans 一样，可以和其他 Beans 一起建立新的 Java 应用。图 1 是 EJB 构件模型的体系结构。

收稿日期：2003-10-21；修返日期：2003-12-07

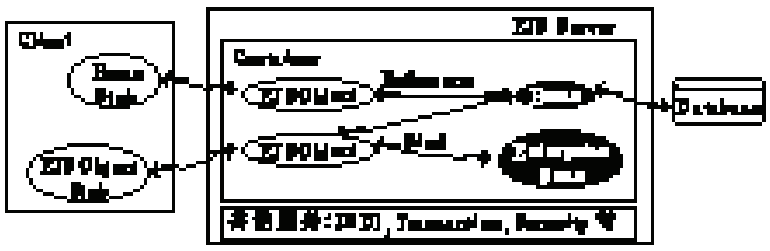

图 1　EJB 体系结构

EJB 构件模型包括 EJBServer, EJBContainer 和 EJBObject 以及诸多相关特性。EJBServer 提供 EJB 应用的运行环境, 它负责管理和协调应用程序资源的分配。EJBServer 必须提供 EJBContainer, 它负责管理这个 EJB 容器, 提供对操作系统服务的存取, 提供 Java 相关的服务。EJBContainer 用于管理 EJB 对象的类, 它负责对象的生命周期的管理, 实现对象的安全性, 协调分布式事务处理, 并负责对象的上下文切换。用户可以通过修改其环境属性来定制 Beans 运行状态特性, EJB 同时支持暂态对象数据及持久对象数据。EJB 对象负责代理用户的调用请求, Home 接口定义一组方法来创建新的 EJB 对象, 查找、定位和清除已有的 EJB 对象, Enterprise Beans 类则是商业逻辑的具体实现类。EJB 2.0 规范中定义了三种 Enterprise Beans: 会话 Beans(Session Beans) 、实体 Beans(Entity Beans) 和消息驱动 Beans(Message- driven Beans) 。会话 Beans 不保存状态信息或数据, 当客户断开连接或服务器关闭时, 会话 Beans 也随之消失; 实体 Beans 模拟商业数据, 表示一个数据存储, 实体 Beans 在客户断开连接或服务器关闭后, 仍有服务保证其数据得以保存; 消息驱动 Beans 在行为上很像会话 Beans, 不同的是仅在需要向这些 Beans 发送消息时才调用消息驱动 Beans。

2.1.3　评价

J2EE 是基于多层、分布式对象体系结构的应用开发。在这种体系结构下, 应用逻辑大部分被从客户端移到服务器端, 分为一个或多个业务对象在应用程序服务器上部署, 这增强了系统的可改进性和可伸缩性, 增加了可靠性、可管理性和灵活性。EJB 技术具有良好的可移植性, 它把 WORA(Write Once, Run Anywhere) 概念提高到一个新的高度。这些构件不仅能在任何平台上运行, 而且它们在任何供应商提供的与 EJB 相容的应用程序服务器之间完全可移植。EJB 环境自动地映射构件到供应商提供的底层执行服务, 它能够与现有基础设施较好地集成。但是多层应用模型的开发难度相对大些, 为了支持多线程、资源共享、复制和负载平衡等, 应用开发难度增大。另外, J2EE 只支持 Java 一种语言。

2.2　CORBA 和 CORBA Component Model

2.2.1　CORBA (Common Object Request Broker Architecture)

CORBA 是由 OMG 提出的应用软件体系结构和对象技术规范, 其核心是一套标准的语言、接口和协议, 以支持异构分布应用程序间的互操作性及独立于平台和编程语言的对象重用。CORBA 的特点可以总结为如下几个方面: ①引入中间件(Middleware) 作为事务代理, 完成客户机(Client) 向服务对象方(Server) 提出的业务请求; ②实现客户与服务对象的完全分开, 客户不需要了解服务对象的实现过程以及具体位置; ③提供软总线机制, 使得在任何环境下、采用任何语言开发的软件只要符合接口规范的定义, 均能够集成到分布式系统中; ④CORBA 要求软件系统采用面向对象的软件实现方法开发应用系统, 实现对象内部细节的完整封装, 保留对象方法的对外接口定义。

CORBA 规范的体系结构模型包括: ①对象请求代理(ORB) 。它使对象能在一个分布式环境下透明地建立和接收请求并应答, 它是建立分布式对象应用的基础, 也是在异构和同构环境中应用之间的交互性的基础。②对象服务(OS) 。它是一个服务集合, 用于支持使用和实现对象的基本功能。③通用设施(CF) 。它是一个面向端用户应用程序的服务集合。

2.2.2　CORBA Component Model(CCM)

1999 年底, CORBA 3.0 提出了服务端构件模型 CCM, 它规范了一个创建即插即用(Plug- and- Play) 对象的框架, 这将有助于集成其他基于对象的技术, 特别是 Java 和 EJB。CCM 的主要贡献是将 CORBA 作为其中间件框架, 标准化了构件的循环开发过程。CCM 采用通用的设计模式(Pattern) , 使得大量代码的产生和系统功能的实现能够通过构件容器(Container) 而不需要应用程序的实现。CCM 是支持多语言的, 它的封装技术比 EJB 更加复杂, 它以 W3C 正在开发的软件封装体系结构为基础来实现。CCM 重点放在服务器端应用程序所需的服务(如 Persistence, Security, Transaction, Event Services 等) , 它把大部分的实现细节留给了构件容器而不是应用程序本身。另外, CCM 支持 EJB 构件, 实现了 Java 语言编写的 EJB 和 CORBA 编程语言编写的 CCM 之间的互操作。

CORBA 构件模型鼓励通过组合而不是继承来重用构件, 为此, CCM 构件可以提供无关的接口(Facet) 并支持客户端的接口导航。图 2 描述了 CORBA 构件模型。

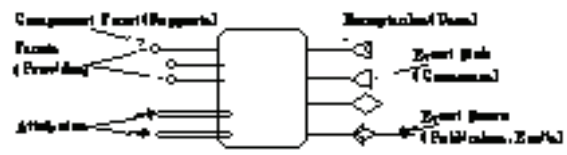

图 2　CCM 模型

在 CCM 中, 构件可以通过称为 Port 的一系列接口与外部实体交互, 这定义了一个调整构件配置的标准机制。Port 机制有四种: ①Facet。Component facet(supports) 使构件可以继承已存在的对象及其方法, 其他 Facets 提供了一些在开发时其他构件也可以使用的接口。②Receptacle 提供了一种通用方式将某种类型的对象和构件关联起来。③Event Source / Sink。构件开发者通过在构件定义中声明事件源和事件槽来发布或预定事件。④Attribute 支持构件属性的配置。

由于构件可以安装到构件 Server 中, 而构件 Server 并没有任何逻辑知识来配置和实例化被安装的构件, 因此, 需要另外的通用接口 Container 来帮助构件 Server 安装和管理它们。CCM Container 编程模型定义了一系列的 API 以简化开发和配置 CORBA 应用, 一个 Container 封装一个构件实现并使用这些 API 向所管理的构件提供运行时的环境。CCM Container 被定义为如何使用底层的 CORBA 基础设施, 它只能在 CORBA 上实现, 因此保证了与其他 CCM Container 的互操作性。CCM Container 使用 CORBA 的 POA 构建了一个能够有效地管理与构件功能实现有关的内存活动框架, 确保 CCM 能够根据需求的增长扩展到企业级而不影响应用程序代码。CCM Container 体系结构图(图 3) 描述的就是 Container 与构件及 CORBA 基础设施之间的关系。

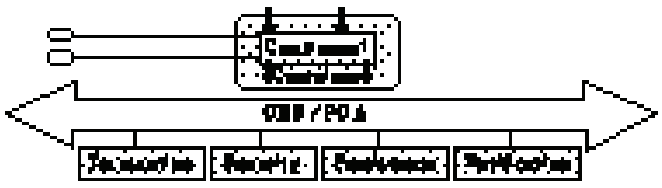

图 3 CCM Container 体系结构图

2.2.3 评价

由于 CORBA 系统引入了中间件的概念,即事务代理,由中间件完成客户机与服务器之间的通信,使得服务器对于客户机的位置相对透明,取消了原有分布式计算模型中客户机、服务器之间的一一对应关系。CORBA 客户机可以在运行时动态获得服务对象的位置,并且可以对多个服务对象提交事务请求,因此极大推动了分布计算的发展。CORBA 规范采用面向对象的分布式软件的构造方法,以接口定义语言的形式实现对象内部细节的完整封装,从而降低了软件系统的复杂程度,增加了软件功能的可重用性。CORBA 提供到 C/C++, Java, Smalltalk 等高级语言的映射,很大程度地减小了对程序设计语言的依赖性,使软件开发人员可以在较大范围内共享已有成果。但是,CORBA 和 CCM 规范庞大而复杂,系统开发比较困难,ORB 的提供商直到最近才开始实现 CCM 规范,就像实现第一代 CORBA 规范一样,现在还很难评估 CCM 实现的质量和性能,而且,不同供应商之间的构件和容器之间的互操作性还没有很好的解释。CCM 规范属于正在发展的规范和技术,仍有待进一步的完善。

2.3 COM/DCOM

2.3.1 COM(Component Object Model)

COM 是一个说明如何建立可动态交替更新的构件的规范,它提供的是客户和构件互操作应遵循的标准。COM 构件是由以 Win32 DLL 或可执行文件(EXE)的形式发布的可执行代码组成的,遵循 COM 标准的构件可以被组合起来以形成应用,COM 构件与语言无关、以二进制形式发布、可以在不妨碍老客户的情况下被升级,并且透明地在网络上被重新分配位置。COM 定义了构件和它们的客户之间互相作用的方式,它使得构件和客户端无需任何中介构件就能相互联系,客户进程直接调用构件中的方法。

客户进程同构件交互的唯一方式就是通过方法调用。客户进程从一个简单的虚函数表(Vtable)中得到这些方法的地址。如果构件是和客户运行在同一个线程中的构件中,那么无需调用任何 COM 或系统代码就可以直接找到方法的地址,COM 仅仅只定义了找到方法地址表的标准。用户和构件不是在同一个线程中时,COM 将它的远程过程调用(RPC)框架代码放到 Vtable 中,然后将每个方法调用打包放到一个标准的缓冲器结构里,这个缓冲器结构将被发送给构件,构件打开包并且重新运行最初的方法调用。从这方面来说,COM 提供了一个面向对象的 RPC 机制。

2.3.2 DCOM(Distributed Component Object Model)

DCOM 基于客户机和服务器模型,扩展了构件对象模型技术(COM),增加了远程调用 COM 构件的功能,使其能够支持在局域网、广域网甚至 Internet 上不同位置的对象之间的通信。当客户进程和构件位于不同的机器时,DCOM 仅仅只是用网络协议来代替本地进程之间的通信。COM 运行库向客户和构件提供了面向对象的服务,并且使用 RPC 和安全机制产生符合 DCOM 线路协议标准的标准网络包。基于 COM 和 DCOM 的设计能使构件在现在和将来都能被很好地使用。图 4 显示了 DCOM 的体系结构。

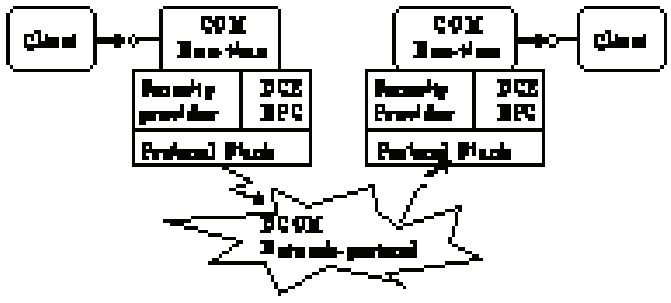

图 4 DCOM 体系结构图

DCOM 选择了 TCP/IP 协议套件中的无连接 UDP 协议作为自己的传输协议。协议的无连接特性使得 DCOM 能够将许多低级别的确认包和实际的数据以及地址合法性检查信息混合起来从而改善了性能。即使是运行在面向连接的协议上,DCOM 也优于传统的面向特殊应用的协议。

2.3.3 评价

DCOM 是一种 ActiveX 技术,所以可以本地化地与 Internet 技术如 TCP/IP, Java 语言和 HTTP 网络协议协同工作,并提供对象接入,使各种商业应用在 Web 上畅通无阻。DCOM 的位置独立性极大地简化了将应用构件分布化的任务,使其能够达到最合适的执行效果。DCOM 是语言中立的,能与用 Java, Delphi, PowerBuilder, C, COBOL, Basic 和 Pascal 等编写的构件在 Internet 上相互通信。DCOM 支持对象编程,明确区分了对象的接口和对象的实现,使开发更加灵活。

COM 和 DCOM 技术为构件和客户进程提供了灵活的扩展机制。一个 COM 构件对不同的客户进程表现为不同的方法和属性组,使用特定特性的客户进程只需要访问它所需要使用的方法和属性,客户进程也能够同时使用一个构件的多个特性。当新的特性加入构件时,它不会影响不涉及这些特性的老的客户进程。

DCOM/COM 的不足之处在于:①依赖于操作系统平台。Microsoft 独家拥有资源(Sole Source Lock-in)。②复杂的对象接口编程风格。由于起源于文档结构技术(OLE),DCOM 使用了复杂的对象接口规范,而不是像 Java 或 Smalltalk 语言那样面向程序设计的风格。为了解决这个问题,Microsoft 提供了一些工具集。③DCOM 的对象定义不如 CORBA 那样直观。

2.4 小结

几种主流的构件技术各自既有所长亦有所短,不同的构件系统开发者应该根据不同的条件和环境来权衡、选择不同的构件技术。表 1 给出了这几种构件技术的综合比较结果。

表 1 主流构件技术比较

		EJB	CORBA/CCM	COM/DCOM
跨平台性		好	好	差
互操作性		一般	好	一般
开放性		好	好	差
支持的开发语言		Java 一种	多种	多种
可扩展性		好	好	一般
开发难度	应用级	易	易	一般
	系统级	较难	较难	一般
产品成熟性		好	一般	好
开发商支持度		好	一般	好
通信机制		RMI	ORB	UDP 协议

3　相关技术对比

3.1　. NET

微软的. NET 战略将互联网本身作为构建新一代操作系统的基础, 并对互联网和操作系统的设计思想进行合理延伸, 使开发人员能够创建出与设备无关的应用程序, 以便轻松实现互联网连接。图 5 是. NET 整体框架图。. NET 在底层的执行引擎也源于虚拟机概念, CLR 不但借鉴了 JVM 的自动垃圾收集、异常处理等机制, 还为. NET 平台添加了多语言支持、构件自描述等新的特性。在程序的编译执行上, 高级语言编译器将 C#及其他. NET 语言翻译成中间语言(IL) (. NET 在这一层面实现了. NET 平台的跨语言承诺), 中间语言被即时编译器(JIT) 编译成特定平台的二进制代码, 完成指令功能。. NET 框架类库提供了一系列的 API, 除去 API 类库的无缝的功能复用外, CLR 也支持本地方法的调用。在异构平台方面, . NET 使用 SOAP。

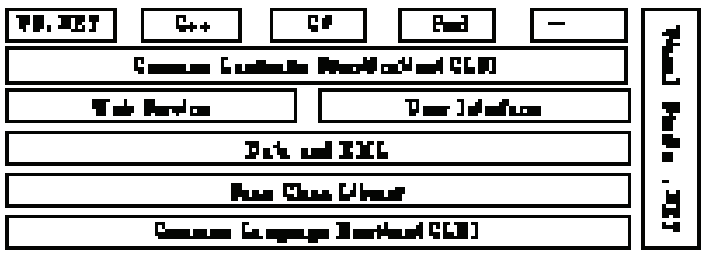

图 5　. NET 框架图

. NET 具有与 J2EE 相似的三层/多层体系, 也分为客户端、中间层和后端数据层, 各层的功能与 J2EE 类似。在移植性方面, 微软通过. NET 通用语言运行时来消除编程语言的差别。J2EE 跨平台不跨语言, 而微软认为平台都是中立的, 希望用跨语言的交互性来平衡跨平台的互操作。J2EE 另一个重要特征就是它的架构开放性, 而. NET 架构的一小部分虽然具有开放性(如 C#语言、通用语言基础构造 CLI 和 Web 服务标准) , 但至少目前很难想像会有一个非微软的. NET 实现。

3.2　开放网格服务体系结构(OGSA)

网格计算是一个正在迅速发展的研究领域, 网格体系结构可以说是网格建设中的核心技术。OGSA(Open Grid Service Architecture) 是当前最新也是最重要的一种网格体系结构形式, 此体系以服务为中心思想, 与构件思想具有一定的可比性。图 6 是网格服务的示意图。

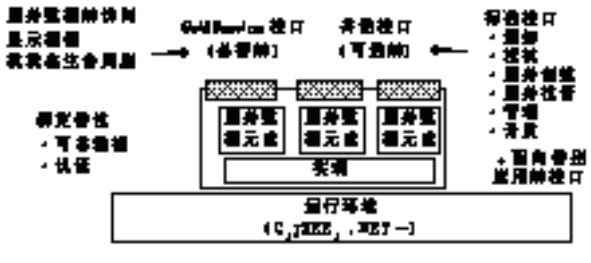

图 6　网格服务示意图

开放网格体系结构 OGSA 是以服务为中心的服务结构, 这里的服务是指具有特定功能的网络化实体。在 OGSA 中, 服务所指的概念包括各种计算资源、存储资源、网络、程序、数据库等, 即一切都是服务。OGSA 的服务模型包括持久服务和临时服务。OGSA 提出了网格服务(Grid Service) 的概念, 用来解决与临时服务有关的服务创建、发现、生命周期管理等问题, OGSA 把整个网络看作是网格服务的可扩展的、动态的集合。

在 OGSA 中, 基本简单的服务能够形成复杂、高级、更抽象的服务, 上面提到的这种服务的包装和服务的组合都与分布式构件的思想比较类似, 对 OGSA 的深入了解更可以发现它从分布式对象技术如 CORBA 演化的痕迹。

4　应用现状与发展前景

4.1　构件技术的应用

(1) J2EE 得到广泛的工业认可。J2EE 是由 SUN 引导, 各厂商共同发起的, 并得到广泛认可的工业标准。与 EJB 兼容的应用程序服务器众多, 包括 BEA WebLogic Tengah, IBM WebSphere Advanced Edition, Novera jBusiness, Oracle 8i, Oracle 应用程序服务器, OrchidSoft Vanda, Persistence PowerTier, Progress Apptivity 等。

(2) CORBA 产品层出不穷。随着 CORBA 规范的推出和不断完善, 世界上有许多计算机供应商和科研人员在对 CORBA 规范进行深入细致的研究和具体的实现。当今基于 CORBA 规范的产品越来越多, 其中比较著名的有: Visigenic 的 VisiBroker, Digital 的 ObjectBroker 以及一些开发源码的项目如 TAO 等。随着网络分布式技术的发展, 基于 CORBA 的产品越来越多地被应用, CORBA 规范将在未来的系统集成和应用开发环境中起到越来越重要的作用。

(3) DCOM/COM 广泛应用。微软的 Windows 开发用户群是微软通过 Windows 操作系统获得的最大财富, 大大小小的公司、研究单位、个人都热衷于开发 Windows 操作系统之上的各种应用, 这有效地促进了 DCOM/COM 的广泛应用。Windows 2000 操作系统本身也是基于 DCOM 开发的成果。

4.2　基于构件的开发

基于构件的开发(Component- Based Development, CBD) 是一种软件开发新方法, 它是在一定构件模型的支持下, 复用构件库中的一个或多个软件构件, 通过组合方法高效率、高质量地构造应用软件系统的过程。

CBD 遵循“购买而不创建(Buy, don ’t build) ”的开发哲学, 让“一切从头开始(Build from scratch) ”的程序编制转向软件组装。基于构件的开发任务包括创建、检索和评价、适配(Adaptation) 、组装、测试和验证、配置和部署(Deployment) 、维护和演化以及遗留系统(Legacy) 的再工程等主要活动。由于以分布式对象为基础的构件实现技术日趋成熟, CBD 已经成为现今软件复用实践的研究热点, 被认为是最具潜力的软件工程发展方向之一。

5　总结

利用分布式构件技术, 我们将实现在分布式异构环境下的软件构件的即插即用功能, 使得现有的应用和系统软件得到合理、高效的重组和使用, 以满足不同的需要。分布式构件技术将在未来的 Internet 技术中发挥重要作用。

参考文献:

[1]　olfgang Emmerich, et al. Component Technologies: JavaBeans, COM, CORBA, RMI, EJB and the CORBA Component (下转第 32 页)