基于消息机制的异构系统集成方案*

彭　竑[1], 赵曦滨[1,2], 雍建平[2], 顾　明[1]

(1. 清华大学 软件学院, 北京 100084; 2. 江苏大学 计算机科学与通信工程学院, 江苏 镇江 212013)

摘　要: Web 服务技术可以为异构系统集成提供有力支持, 但在异步调用、流程控制和事务处理等方面仍然存在许多困难。提出了基于消息机制的异构系统集成方案, 设计了总线型消息控制平台, 通过基于 XML 的格式化消息提升系统集成的能力, 并在数字化校园建设中得到了应用。

关键词: 消息; 异构系统; Web 服务

中图法分类号: TP393　　**文献标识码**: A　　**文章编号**: 1001-3695(2005)08-0043-04

Heterogeneous System Integration Solution Based on Message Mechanism

PENG Hong[1], ZHAO Xi-bin[1,2], YONG Jian-ping[2], GU Ming[1]

(1. *School of Software, Tsinghua University, Beijing* 100084, *China*; 2. *School of Computer Science & Telecommunications Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*)

Abstract: Heterogeneous system integration has an important meaning to impulse further development of information of enterprises. Web services technology can give forcefully support for heterogenous system integration, but in some aspects there are still many difficulties such as asynchronous invoke, business process control and transaction, etc. Presents a heterogenous system integration solution based on message mechanism, designs a bus model of message control platform, using the formatted message by XML to improve the ability of system integration. The solution is practised in the construction of digital compus.

Key words: Message; Heterogeneous System; Web Services

1　引言

Internet 的飞速发展对现代生产和生活方式产生了深远影响。众多面向政府办公、商业贸易和生产制造的应用系统都在借助网络获取和发布信息。随着网络应用的深入发展, 对应用系统内部和系统之间的信息共享与协同工作也提出了更高的要求。如何更好地完成基于异构系统的应用集成已成为理论研究和工程实践的热点问题。传统的信息系统开发很少考虑应用的可集成性、可重用性、可定制性和可移植性, 这就造成了众多软/硬件平台及各类应用系统并存的局面, 在实施信息化的同时形成了新的"信息孤岛"。当众多系统需要进行数据交换和流程组合时, 目前的解决方案基本上是以某一个或某几个关键应用系统为主, 基于系统提供的接口进行二次开发, 与其他系统进行整合。这是一种紧密耦合的系统集成模式, 它依赖于特定的编程接口和开发平台, 其接口的复杂性及在不同技术供应商之间进行工作协调的困难使企业望而生畏, 给系统集成带来了许多麻烦。此外, 随着企业信息化的进展, 新的应用部署需要考虑新的接口开发, 应用层面发生的更改常常导致众多复杂接口的升级, 面向异构系统的集成将变得越来越困难[1]。

企业应用集成 (Enterprise Application Integration, EAI) 等解决方案被提出用于解决上述问题。EAI 的目标是通过硬件、软件、标准和业务过程的结合, 实现两个或多个企业系统之间的无缝集成, 使它们能够统一运作[2]。当前一些主要的 EAI 解决方案采用了 COM+/CORBA/Java RMI/EJB 等组件对象技术, 并且在金融、电信等领域中取得了广泛的应用。但是, 上述解决方案在实施时面临着许多困难。①来自不同供应商的组件及采用不同技术的组件之间的互操作性给异构系统集成造成了很大困难。每个组件模型都希望被作为业界所接受的唯一标准, 但这在 Internet/Intranet 时代中是不切实际的, 无论从商业利益还是学术研究的角度来看, 组件模型的多样化都是必然的结果。此外, 即使是采用相同的组件模型, 也会因为由不同的软件厂商实现或是因为使用的组件模型的版本不同而导致沟通困难, 这就是所谓的互操作问题。②当前诸多的应用系统出于安全性考虑, 都在系统外面部署了防火墙, 只允许 HTTP 包通过, 传统的组件模型虽然采用种种技术解决防火墙问题, 但是都显得过于复杂。③目前的组件大多只能代表细粒度的软件逻辑, 而无法封装粗粒度的应用逻辑, 这也给应用级的重用带来了困难, 不利于提高系统的兼容性和可扩展性。

针对上述问题, Web 服务 (Web Services) 提供了新的异构系统集成思路。Web 服务体系打破了传统系统集成方法造成的紧耦合局面, 采用 Web Services 来进行新系统开发或对原有系统进行封装, 因而保持了各个异构系统的自治性。Web Services 之间遵循简单对象访问协议 (Simple Object Access Protocol, SOAP), 并采用 XML 作为数据交换标准, 在统一数据模型的基础上连接了数据孤岛, 实现了松散耦合的系统集成。此外, 由于 SOAP 协议建立在 HTTP 基础之上, 因而在实施时不会因防火墙设置而产生太大的困难。

尽管较传统的 EAI 解决方案而言, Web 服务技术提供了更好的交互能力和集成效率, 但其解决基于复杂业务流程的异

收稿日期: 2004-06-26; 修返日期: 2004-11-23

基金项目: 国家"863"计划资助项目(2003AA414031, 2003AA413031)

步调用和事务处理时也面临很大的困难。

2 Web 服务在异构系统集成中应用分析

Web 服务是自包含、自描述、模块化的应用,可以在网络中被描述、发布、查找以及通过网络来调用。它向外界暴露出一个能够通过 Web 进行调用的 API,使得所有应用只需要能够连入 Internet,就可以使用和集成 Web 服务。Web 服务能够为企业应用集成提供有力的支持。借助 Web 服务技术可以在企业内部和企业与企业之间实现各种应用的相互调用,以实现面向应用的资源共享与协同工作。目前,基于 Web 服务的企业集成应用方案得到了越来越广泛的应用。但是,在实施基于 Web 服务的异构系统集成时仍然存在许多困难。

(1) 在 Web 服务体系中,服务的请求及获取都借助 SOAP 来进行,而 SOAP 遵循同步工作模式,也即服务请求者发送 SOAP 请求后,在服务提供者进行处理的那段时间内,服务请求者一直等待 Web 服务的处理结果。一般来说,如果不作任何处理,服务请求者发送 SOAP 请求和接收 SOAP 响应是在同一个线程执行的。也就是说该服务请求者在服务提供者进行服务处理的过程中一直是闲置的。此外,由于客户端对远程方法调用的依赖性,一旦系统出现了一个同步调用失败,调用的应用程序将发生崩溃或出现不可预测的后果。显然,这种模式必将降低系统的灵活性,削弱系统的健壮性。

(2) 采用 Web Services 可封装原有的各个业务子系统,使得这些系统可提供对外的服务接口。但是如果想让它们有机地结合起来,保持相互间的协调,并按照一定的流程来共同推进一个业务的完成,仅仅封装原有的业务系统还是不够的。企业信息系统集成的出现正是由于原有孤立系统不足以完成某项业务,故而需要这些异构子系统能够协同工作。由于异构业务子系统是相互独立,如果业务仅仅是让这些业务子系统依次处理则很容易实现,而实际情况往往要复杂很多。Web 服务的消费者和提供者都需要保持各自的独立活动,这必然给协作同步带来困难。此外,如果这些业务子系统中途崩溃,或者处理结果返回的是异常乃至错误,其他的业务流程如何进行下去,也是系统集成中所必须考虑的重点问题。产生上述问题的一个重要原因在于 Web 服务是松散耦合的,彼此之间没有逻辑上的衔接和从属关系,从而缺少有力的流程控制。

事务处理也是 Web 服务在集成应用时的一个难点。事务往往被称为“逻辑工作单元”(Logical Unit of Work, LUW),是将数据从一种一致的状态转换为另一种状态的一组操作。事务处理的概念最早在数据库系统中被提出,指的是一组可靠的执行和对数据的操作,它们或者是全部的执行,或者是全部不执行,用来保护中央数据库中的数据。随着分布式计算的发展,事务在分布式计算领域中也得到了广泛的应用。分布式事务是指事务的参与者、支持事务的服务器、资源服务器以及事务管理器分别位于不同的分布式系统的不同节点之上,这样一个事务可能涉及多个数据库操作。分布式事务处理的关键是必须有一种方法可以知道事务在任何地方所做的所有动作,事务的提交或回滚必须产生统一的结果(全部提交或全部回滚)。

目前常用的分布式事务处理的方式是基于 X\Open 的模型,它需要支持一套给定的接口,参与分布式事务的各个分支之间需要支持两段式协议,从而能够统一地进行事务的提交和回滚。大部分的交易中间件都是基于这样的模型。但是,这些中间件的价格都比较昂贵,此外,很多老系统并不能支持两段式的协议。尤为重要的是,各个业务子系统通过封装为 Web 服务后,Web 服务标准目前并不支持两段提交协议[3]。而目前的 Web 服务中事务处理标准也在研究和规范中。这样,通过 Web 服务封装的异构系统之间的衔接与协作来共同推进完成一个事务就遇到了很大的困难。

综上所述,Web Services 提供了有统一的架构和技术规范,使得用户能够以平台独立和语言独立的方式连接不同的异构系统,从而使得异构系统的集成大大地向前迈进了一步。但是这些标准并没有定义 Web 服务的业务语义,采用 Web 服务标准封装后的异构系统还是彼此孤立,需要通过一种控制机制能够使它们连接起来一起完成业务,同时能够更好地解决异步消息和事务处理问题,这样才能更为有效地完成异构系统的集成。

3 基于消息机制的异构系统集成方案

3.1 平台的架构设计

如上节所述,仅仅将原有的异构系统封装为 Web 服务并不能保证协同工作的顺利进行。为此,本文提出在 Web 服务之上构建一层总线型消息控制平台。这样的总线型控制平台能够接收外界用户的业务请求,通过进行业务解析将复杂的业务分解为业务流程描述和由各个业务子系统提供服务而完成的业务子操作。总线型控制平台根据业务流程采用格式化的消息来向业务子系统提出服务请求,调度它们来完成业务子操作,并根据业务子系统返回的结果进行协调控制,最终将业务处理结果返回给用户。其平台的架构设计如图 1 所示。

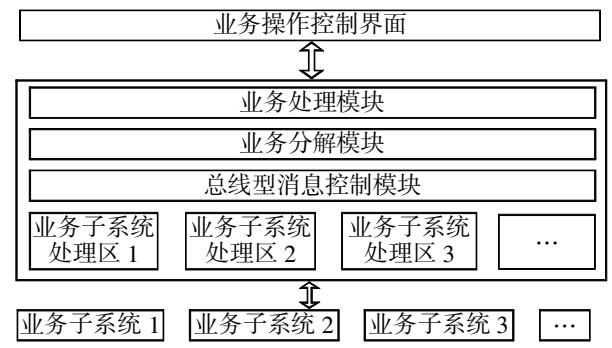

图 1 总线型消息控制平台架构图

图 1 中的各业务子系统是将原有的异构子系统按照 Web 服务模式所封装起来的。这些业务子系统能够提供服务并完成一定的业务操作。总线型消息控制平台构建在这些异构的业务子系统之上。对外界提供业务操作控制界面,业务用户通过浏览器可以很容易地进行访问和操作。用户提交业务操作的请求并将控制要求将传送到总线型消息控制平台,由它来完成相应的调度执行。总线型消息控制平台由三个主要模块构成:

(1) 业务处理模块。用户提交的业务操作请求将被这个模块所接受。业务处理模块缓存待处理的业务请求消息,并将该业务请求发送到业务解析模块。最后业务处理的结果传到业务处理模块中,与先前的业务操作请求相对照,最终将结果返回给用户。

(2) 业务解析模块。它负责接收由业务处理模块发送过来的业务请求,将业务分解成业务流程逻辑和一系列由各个业务子系统完成的业务子操作。业务流程逻辑中描述了这些子

操作的协调调度关系。这些业务子操作和业务流程将发送到总线型消息控制模块。

(3) 总线型消息控制模块。它处于核心的位置。根据业务解析模块分解的业务流程逻辑统一发送消息调用各个业务子系统提供的业务子操作, 并根据处理的结果进行协调处理。在所有的业务子操作都成功完成后, 总线型控制模块将这些业务子系统的处理结果汇总为业务处理结果, 并将其反馈到业务处理模块, 由业务处理模块将最终结果反馈给用户。在业务流程中服务处理也可能出现异常或者失败, 总线型控制模块能够进行相应的业务流程逻辑进行控制, 重新发出服务请求的消息, 或者由于某种原因无法继续时, 向业务处理模块返回业务处理失败的消息。

此外, 在总线型消息控制平台中还为每个业务子系统设置了一个相应的调度处理区。在这个处理区中, 设立专门的队列, 从总线控制模块接收到相应的业务子操作调用消息将转存到这些队列中。根据这些调用消息向业务子系统发出服务调用请求。业务子系统在接到调用后发送调用确认消息, 业务处理区在收到这个确认消息后, 将相应的调用消息转移到处理区中的另一个队列, 这个队列缓存在处理进行中的调用请求, 等待作为服务提供者的业务子系统返回服务处理结果。在业务子系统处理结束后接受处理的结果, 并同该调用相匹配后反馈给总线型消息控制模块。总线型消息控制模块汇总最后的处理结果发送到业务层处理模块。

3.2　消息处理机制

消息服务系统允许未耦合的应用程序之间可靠地进行异步通信, 即发送方将消息发送到一条虚拟通道(主题或者队列)上, 不在乎是否马上能得到回应。这样, 发送方无需等待接收方处理请求, 只需确保消息成功发送后, 就可以处理其他任务, 从而极大地提高了系统的效率。消息接收方通过订阅或是监听该通道, 也无需监视接收的全过程, 只需要简单地读取消息并处理。一条消息可能最终转发到一个或者多个接收方。消息系统提供了许多其他分布式对象计算模型没有的优点。它支持消息产生者(Message Producer) 和消息使用者(Message Consumer) 之间的松散耦合, 在它们之间有很高程度的事务处理。使用者不必关心是谁产生了消息、产生者是否仍在网络上以及消息是什么时候产生的。整个子系统能被修改而不会影响系统其他的部分, 从而可以建立动态的、可靠的和灵活的系统[4]。

消息队列可以允许不同的应用程序之间进行信息交流。应用程序能够同步或异步地向消息队列中发送或接收消息, 可以形象地把消息看作应用程序的电子邮件。为消息队列开发的应用程序可以将消息发送到队列, 这些队列是用来确保消息能够到达目标的存储位置[5]。这些应用程序可以通过不同种类的网络进行通信, 也可以与脱机的计算机通信。消息队列提供了有保障的消息传递、有效的路由、安全性、事务支持以及基于优先级的消息传递。消息机制的优点在于能够在客户和服务之间提供同步和异步的连接, 并且在任何时刻都可以将消息进行传送或者存储转发, 这也是它比远程过程调用更进一步的原因。

目前的消息系统如 JMS, MQ 等一般将消息的内容视为二进制字节流, 仅仅通过消息头的少量信息来进行转发控制, 而对于消息的内容和含义不作任何区分。但是实际上消息可能有一定的分类和含义。需要通过制定出一定的格式来加以区分, 从而能够有效地控制于不同的异构系统。比如说, 发出服务请求调用的消息可能需要包含服务方的服务标志和参数, 而结果反馈的消息则需要包含调用请求的标志和结果数据。考虑服务质量, 还需要在控制消息中设置安全标志, 服务时间等项。消息的格式化可以通过 XML 来完成。Web Services 在服务提供者与调用者之间就是通过发送基于 SOAP 的 XML 文件来调用的。可以将这样的调用机制扩展应用到消息机制中。

在总线型消息控制平台中, 就是通过接收和发送用 XML 格式化的消息来协调调度各个异构的子系统的。通过格式化可以将消息区分为业务事件、服务请求消息、请求确认消息、结果反馈信息、异常消息等不同的类型。在业务子系统区和业务层处理模块中设置了一系列的队列, 用来存储转发各个系统之间的异步消息。通过对消息的控制来协调控制异构系统完成业务请求。下面是一个这样格式化消息的实例:

```
<?xml version="1.0", encoding="UTF-8" >
<message>
  <header>
    <messageID>
      BPFT98038202
    </messageID>
  <status>
    <affirminvoke>
      <invokemessageID>
      BPFT98038201
      </invokemessageID>
    </affirminvoke>
    <time>
    2004-6-21-9:56-25
    </time>
  </status>
    <receiver>
      MessageControlPlatForm
    </receiver>
    <sender>
      <provider>
        LibraryService
      </provider>
    </sender>
  </header>
</message>
```

上例是在总线型消息控制平台向图书馆系统发出服务请求, 图书馆系统的 Web 服务在收到该请求后给出的确认信息。该消息头部给出了所确认的服务请求消息的消息号和时间戳。随后给出了此项消息的接收者和发送者, 分别为总线型消息控制平台和图书馆 Web 服务系统。最后图书馆的 Web 服务将处理此项业务请求。在完成这项业务子操作后, 将服务处理结果封装在消息中再返回给总线消息控制平台。

3.3　总线型消息控制平台的操作流程

假设业务操作涉及到三个业务子系统的子操作。业务子操作 3 需要在业务子操作 1 和业务子操作 2 结束后才能根据相应的结果发出调用。总线型消息控制平台将首先向业务子系统 1 和业务子系统 2 发出服务调用请求。业务子系统 1、2 都先返回服务请求确认的消息, 然后分别进行相应的服务处理, 最后将各自的处理结果返回给总线控制平台。总线型控制平台确认这两个服务调用成功返回处理结果后, 再向业务子系统 3 发出服务请求调用。最后在业务子系统 3 处理结束后将结果返回到总线型消息控制平台中。由总线型消息控制平台汇总最终的业务处理结果返回给用户。其顺序如图 2 所示。

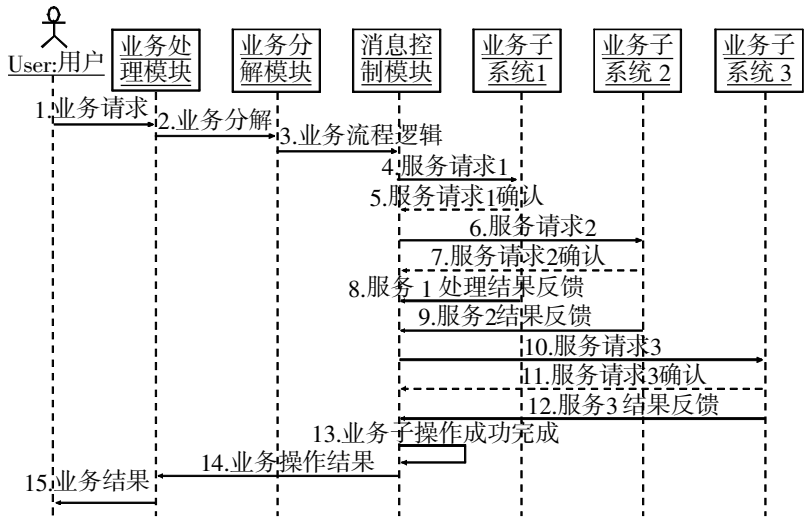

图 2 总线型消息控制平台流程图

考虑服务处理出错的时候处理流程的变化。假设在处理过程中,业务子系统 2 由于某种原因处理服务请求失败,并且返回了处理失败的消息。此时根据业务流程逻辑不能再继续去调用业务子系统 3。总线型消息控制平台将根据业务子系统 2 返回的失败结果,重新向它发出服务调用请求信息,等待业务子系统 2 返回正确的处理结果后再继续下一步业务流程,向业务子系统 3 发出服务调用请求。

如果业务子系统 1 和子系统 2 的操作在一个分布式的事务中,此时由于业务子系统 2 处理失败,总线型控制平台需要发送信息给业务子系统 1,通知它进行事务回滚的操作。此后需要重新启动事务,发送服务调用消息让业务子系统 1 和子系统 2 进行处理。如果业务子系统 2 由于某种原因崩溃而不能再提供服务,则总线型控制平台将中止业务流程,发送消息通知业务子系统 1 回滚事务,并给用户返回业务请求失败的信息。

通过构建总线型消息控制平台能够很好地解决前面所分析现有 Web 服务体系中的问题:①消息机制能够很好地解决异步处理问题。通过建立一系列消息队列,能够收发异步的消息,这使得服务请求者不必再等候服务的结果,服务提供者可以在处理结束后异步地将服务结果发送回来。②业务解析模块根据业务请求能产生相应的业务流程逻辑。总线型消息控制平台根据这个业务流程进行协调调度。从而能够使得异构系统协同工作以完成业务。③根据业务流程逻辑采用消息机制还可较好地解决 Web 服务模型下分布式事务处理的问题。总线型消息控制平台根据业务逻辑启动事务,并通过发送消息启动有关的业务子系统的分支事务。在某个分支事务处理失败后,总线控制平台将发送消息让其余分支事务回滚。此后重新启动分布式事务。

4 应用实例

本文提出的解决方案在某高校数字化校园建设中得到了应用。该高校原来陆续引入过一些软件系统,如财务系统、图书馆借阅系统、后勤饭卡系统和学生注册系统等。但是这些系统都是不同的时期,用不同的开发平台开发于不同的系统之下的。以前运行中这些系统各不相关,给学校管理人员带来很大的不便,不能充分发挥信息化的优势。比如学生退学的时候,可能还有图书没有还,而在后勤系统中还有一定的存款等。这在以往都需要工作人员和学生采用很烦琐的手续,有所疏忽则会带来数据有所不一致的冗余。学校希望能够集成现有的系统,提供统一的操作平台,发挥信息化的优势,因而展开了数字化校园的建设工作。

数字化校园是运用计算机、网络、通信等先进的信息化手段和工具,将学校的教学、科研、管理和服务等有关信息资源进行全面的数字化整合。现阶段校园网应用最大的问题是以往的业务系统都是以应用为中心,完成特定功能的子系统都是封闭的,部署在不同的平台上,利用了不同的数据库系统进行数据存储,采用不同的开发技术进行系统的开发。这些异构性造成了各个系统之间的通信和数据交换,共享困难,数据冗余度很大。各行其是的管理带来很大的混乱。数据在其中可能有很多的不一致性。需要整合老的异构系统,开发出新的信息化平台。实际应用中将现有的各个系统的各个子系统应该被看作 Web 服务体系结构中的服务提供者的角色,所有的接口信息都以网络服务描述语言(Web Services Description Language, WSDL)的统一形式展现给上层的系统模块。而在这些 Web 服务之上,我们提供统一的总线型的消息控制平台来协调调度这些系统的操作。数字化校园体系结构图如图 3 所示。

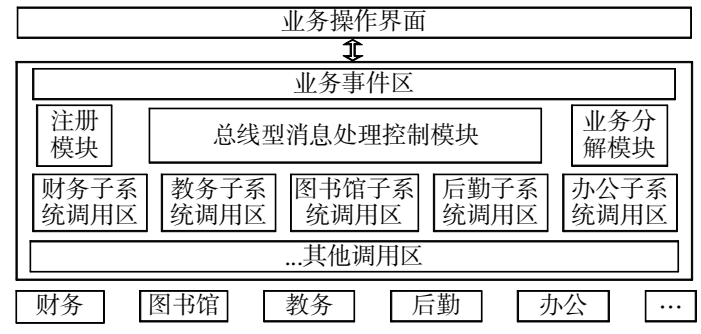

图 3 数字化校园体系结构图

5 结论

本文分析了异构系统集成的产生和传统中间件解决方案的缺点,介绍了 Web Services 技术体系以及它在异构系统集成方面的优势。同时也分析了目前 Web Services 标准下的一些不足之处,如在异步处理、业务协同、事务处理方面的困难。通过将消息机制引入其中,在已经由 Web 服务标准封装好的各个业务子系统之上再构建一个总线型消息控制平台。从用户获得业务请求,通过业务解析模块分解产生相应的控制信息,根据这些请求信息调动各个业务子系统协同完成业务。同时能灵活地解决异步处理、事务和业务流程控制等方面的问题,并在具体的工程中的到了应用。进一步的研究将关注于标准业务流程语言及其在 Web 服务中的应用,以及如何将消息机制和事务处理关联起来,以更好地推动 Web 服务在异构集成系统中的应用。

参考文献:

[1] 许科峰,高建民,陈富民,等. 基于 Web Services 的企业应用集成技术及实现[J]. 计算机应用,2004,24(12):155-157.

[2] 柴晓路. Web 服务构架与开放互操作技术[M]. 北京:清华大学出版社,2002.

[3] James Snell. Web 服务内幕,第 10 部分:深入主题:可靠性和事务[EB/OL]. http://www.huihoo.com/xml/ws-ref/index10.html.

[4] 纪波林,王志坚. 基于 JMS 体系结构的消息服务技术的应用研究[J]. 计算机应用研究,2003,20(11):48-51.

[5] 卢本捷,魏守平,刘俊. 基于消息队列中间件的总线式工作流管理系统设计[J]. 计算机应用研究,2004,21(1):159-161.

作者简介:

彭竑,男,硕士研究生,主要研究方向为电子商务、企业应用集成,Web 服务;赵曦滨,男,博士研究生,主要研究方向为计算机系统安全、软件工程;雍建平,男,讲师,主要研究方向为数据库技术与软件工程;顾明,女,副教授,主要研究方向为操作系统、中间件技术、分布式应用系统支撑平台、电子商务等。