远程门户组件的 Web 服务的研究与实现*

赵　勇, 马殿富
(北京航空航天大学 计算机学院, 北京 100083)

摘　要: Web 服务解决了异构环境中的互操作问题,但是并没有提供信息展示和用户交互的方法。采用远程门户组件的 Web Service (WSRP) 技术,提出了 WSRP 平台的概念模型,分析了远程门户组件的映射原理,研究了远程门户组件的对象池及管理算法,设计实现了一个远程门户组件的 Web 服务的运行平台,解决了 Web 服务的展现问题,同时解决了 Internert 上的门户组件共享问题。

关键词: 远程门户组件的 Web 服务; 门户组件; 生产者; 消费者
中图法分类号: TP393.09　　**文献标识码**: A　　**文章编号**: 1001-3695(2004)11-0247-04

Research and Implementation of Platform of Web Services for Remote Portlet

ZHAO Yong, MA Dian-fu
(School of Computer, Beijing University of Aeronautics & Astronautics, Beijing 100083, China)

Abstract: Web Services provides an effective solution for the interoperability and integration of heterogeneous applications. However, it can not handle service presentation and user interaction. Introduces the technique of Web Services for Remote Portlets(WSRP), brings a conception model of WSRP platform. Based on the WSRP satandard specification, analyses the mapping principle of remote portlets, researches the arithmetic of remote portlets object pool. A layed platform of Web Services for remote portlet is designed and implemented, solving the problem in presentation of Web Services and sharing of portlets in Internet.

Key words: Web Services for Remote Portlets(WSRP); Portlet; Producer; Consumer

目前 Internet 的应用技术已经从简单的信息浏览发展到了复杂的分布式应用, Web 服务的出现加速了现有应用的集成,方便了企业与企业之间的互操作,成为下一代软件服务的基础[1]。Web 服务虽然解决了机器与机器之间的互操作问题,但是它本身不包括任何用户交互或表示功能,因此需要中间应用程序来提供用户界面。这就需要为每个 Web 服务进行特定的编码来展现,在实际应用中,就会带来应用周期长、重复编码等问题[2]。同时,随着企业信息门户技术的发展,不同的门户平台之间门户组件的互操作性成为日益突出的问题[3]。目前存在很多的门户平台,这些平台中的门户组件使用不同的接口,互相不具有兼容性。在 Web 服务的展现方面,近年来已经取得了一些研究成果[2,9]。Takeshi 等人[9]提出一种基于 Web 表单的 Web 服务界面表示方法,而 WSUI[8] 通过一个 Schema 来描述出专门的 Web 服务用户层,并使用 XSLT 样式单来构建用户界面,限定了界面元素,缺乏灵活性。而 WSIA[2] 依赖于 W3C 的 XForm 的表现技术和语义网技术来构建 Web 服务界面,实现过于复杂。只有 WSRP(Web Services for Remote Portlets) 使用已经成熟的 Portlet 技术来构建 Web 服务界面,同时解决了门户组件的跨平台重用问题,因此成为我们关注的重点。

WSRP 是 OASIS 组织在 2003 年 9 月提出的一项开放标准[5,6],位于 Web 服务协议栈的最顶层。WSRP 利用已有的 Web 服务标准,使用 WSDL 定义了一个 Web 服务接口描述,使用标准的 Web 服务接口包装远程门户组件。这些接口独立于具体的应用,只描述了如何获取使用远程门户组件,保证了调用者如何与远程门户组件交互,而不规定组件的具体实现,因此具有良好的互操作性,并且使得 WSRP 服务具有可视化、面向用户、即插即用的重要特点,将极大地方便 Web 应用程序的快速集成。但是如何实现 WSRP 接口,利用规范的开放标准,实现信息共享和真正的互操作,如何提供一种基础结构能够容纳不同接口的远程门户组件等仍是当前 WSRP 应用中的重要问题。已有的系统大多基于对现有门户平台的改造,实现门户组件的远程使用,而不是针对远程门户组件的特殊情况。本文在分析了 WSRP 原理的基础上,给出了独立于门户平台的远程门户组件 Web 服务运行平台的概念模型,研究了远程组件的映射原理,并设计实现了一个远程门户组件的 Web 服务的运行平台,实现了远程门户组件 Web 服务的统一处理机制。

1　WSRP 技术研究

1.1　门户组件的运行原理

门户是对应用的单一集成访问点,它可以集成不同来源的信息和应用,为用户提供完整的应用视图。门户的构建基于一系列的门户组件。门户组件是一种门户服务器端的组件,它实现业务功能,产生面向用户的可视化的标记片段(如 HTML 标记片断),这些标记片段经过门户组件整合模块的整合,成为最终用户页面(图 1)。门户组件的输出可能使用到了数据库、文件或者 Web Services 等数据源。

不同的平台,其组件接口和实现方式是不一样的,如 IBM WebSphere Portal Server 采用 Servlet 接口实现门户组件,而 A-

收稿日期: 2003-11-09; 修返日期: 2003-12-29
基金项目: 国家“863”计划资助项目(2001AA116050, 2001AA110485, 2001AA144150)

pache Jetspeed 使用了自己定义的 Java 接口, 不同的门户组件之间的接口不能兼容。为了可以在 Internet 上共享远程门户组件, 就产生了远程门户组件的 Web 服务即 WSRP 技术。WSRP 技术借助 Portlet 的展现能力实现了 Web 服务的用户界面, 并且实现了不同平台上的门户组件的共享。

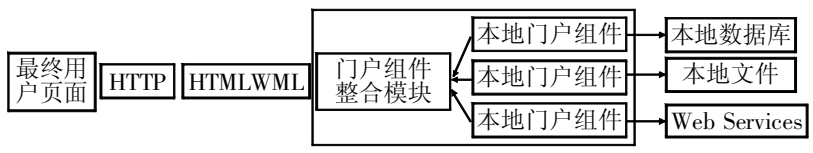

图 1 门户系统构成

1.2 WSRP 的技术概述

WSRP 使用 Web 服务技术将门户组件用标准的接口包装, 规范定义了这些接口的详细描述, 并说明在 WSRP 中主要使用生产者、消费者这两种角色。消费者是远程组件的使用者, 可能是一个门户网站, 或者是其他的 Web 应用。生产者通过 Web 服务接口向消费者提供远程门户组件。接口满足标准的 WSDL Schema 定义。

生产者对外表现是一个 Web 服务, 消费者的任务是通过生产者 Web 服务接口获取远程门户组件提供的内容。因为所有生产者的接口一致, 所有消费者可以方便地重复消费远程门户组件, 也就是一次编码, 到处调用。这些接口包括:

(1) 消费者注册接口。消费者向生产者注册, 并可以修改注册信息。注册接口包含三个操作, 即注册, 消费者向生产者注册, 形成一种注册关系, 准备使用远程门户组件; 取消注册, 消费者不再使用生产者的远程门户组件, 就需要取消注册关系; 修改注册, 消费者修改在生产者处存放的注册信息。

(2) 生产者服务描述接口。消费者动态地获取生产者提供的门户组件详细信息, 以决定如何使用这些远程门户组件。描述接口包含一个操作获取服务描述以得到生产者的详细信息。

(3) 远程组件定制和配置接口。消费者可以向生产者要求新的可配置的组件实例, 并配置这些组件的属性, 使得不同的消费者使用不同的组件实例。本接口最重要的操作为复制实例的操作, 当消费者获取远程门户组件实例并配置其属性以后, 就在消费者页面中增加了远程门户组件的映射。

(4) 标记片段和用户交互接口。消费者通过该 Web 服务接口可以动态得到远程门户组件的标记片断, 并与之交互。这个接口是 WSRP 技术的关键, 因为 WSRP 服务可以返回 HTML 标记片断, 因此就不需要客户端再增加显示逻辑。

门户消费者其最终页面的输出可能包含远程组件的内容, 也可能包含本地组件的内容。对于一个生产者而言, 需要将远程门户组件提供给许多不同的门户消费。一个典型的应用场景如图 2 所示。

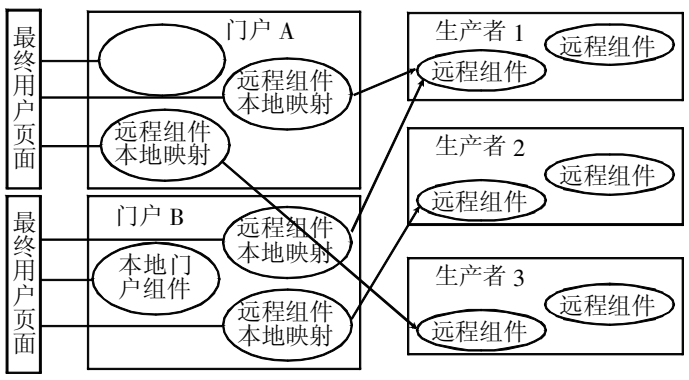

图 2 WSRP 应用场景图示

2 系统设计原理

系统设计的关键问题在于: 生产者如何管理维护大量的远程组件供消费者使用, 如何支持现有的多种门户组件, 保证兼容性; 消费者如何通过标准的接口调用远程门户组件, 将其融合到门户页面中。我们首先抽象出系统的概念模型。

2.1 概念模型

定义 1 Ir = (Or, Odr, Omr) , 表示注册接口的操作集合, 包括注册、取消注册、修改注册三个操作。

定义 2 Id = (Osd) , 表示服务描述接口, 包含获取服务描述这一个操作。

定义 3 Ic = (Ogp, Ocp, Osp, Ogpp, Oppd, Odp) , 表示组件管理接口, 包含六个操作, 分别是获取组件描述、复制组件实例、设置组件属性、获取组件属性、获取属性描述和销毁组件实例。

定义 4 Im = (Ogm, Opb, Oic, Ors) , 标记接口的操作集合, 包括获取标记、执行交互、初始化和释放会话。

定义 5 I = Ir ∪ Id ∪ Ic ∪ Im, 表示 WSRP 接口的集合, 分为以上四种接口类型。

定义 6 P = { Ci, Ogm (Ci) = HTMLfrgment} , 表示生产者提供的远程门户组件的集合, 其中远程门户组件标记操作输出为 HTML 标记片段。

基于定义 1 ~定义 6, 生产者可以描述为一个二元组 Producer = (I, P) , 表示生产者实现标准接口, 提供远程门户组件。

对于一个使用远程门户组件的消费者, 可以抽象出四元组作为其模型描述, 表示为 Consumer = (Url, Ph, Pl, Prm) , 其中, Url 是一个消费者的地址, 是最终用户通过浏览器访问的路径; Ph 是消费者向最终用户提供的页面, 这些页面是由门户组件的输出集成而成的 HTML 页面; Pl 表示本地门户组件; Prm 表示远程门户组件在本地的映射, 是一种特殊的本地门户组件。

对于门户而言, Ph = O(Pl) + O(Prm) 。也就是说, 门户页面由本地和远程门户组件的输出集成表示。如果我们可以使得本地组件的输出和远程组件一致, 就完成了一个本地组件到远程门户组件的映射。因此需要 O(Prm) = Ogm(P) , 其中 P 即为定义 6 中的远程门户组件集合。

生产者表现为一个 Web 服务, 而消费者就是该 Web 服务的使用者, 因此消费者首先要向 UDDI 获取生产者的地址和元信息, 然后才能使用生产者。当消费者发现了生产者 Web 服务以后, 所有消费远程门户组件的工作都是通过这四组接口的操作完成。在交互过程中, 会产生不同的状态, 我们使用一个状态转换图来描述这种关系, 如图 3 所示。

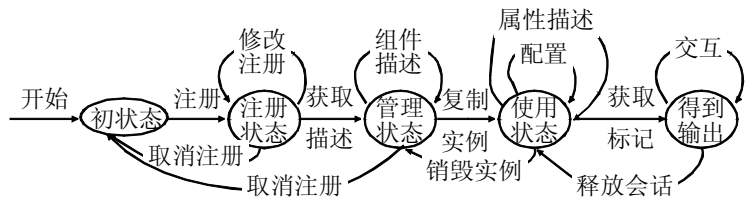

图 3 状态转换图

消费者在使用过程中存在以下几个状态:

(1) 初始状态 S_0。消费者准备消费生产者的门户组件, 首先需要向生产者注册

f(S_0, Or) = Sr; 注册后会跳转到注册状态 Sr

(2) 注册状态 Sr 。表示消费者已经向生产者注册, 等待下一步操作, 在该状态下的转移包括

f(Sr, Odr) = S_0 取消注册关系, 回到初始状态

f(Sr, Omr) = Sr 修改注册信息

f(Sr, Osd) = Sm 获取生产者的详细描述, 进入管理状态

(3) 管理状态 *Sm*。在该状态下的转移包括

f(Sm, Ogp) = Sr　获取某个门户组件的具体信息

f(Smr, Ocp) = Su　复制一个远程门户组件的实例, 进入使用状态使用之

f(Smr, Odr) = S$_0$　取消注册关系, 回到初始状态

(4) 使用状态 *Su*。在该状态下的转移包括

f(Su, Oppd) = Su　得到属性描述

f(Su, Osp) = Su　配置该实例的属性

f(Su, Odp) = Su　销毁该组件实例

f(Su, Ogm) = So　获取标记, 转向得到输出状态, 这里的标记就是远程门户组件的输出

(5) 输出状态 *So*。当最终用户通过消费者获取到远程门户组件的输出后, 就与远程门户组件建立会话关系, 可以与之交互。

f(So, Opb) = So　执行交互后仍然要获取标记

f(So, Ors) = Su　释放会话回到使用状态

以上的状态转移就包括了消费者和生产者之间通过 WSRP 接口交互的过程, 在实际应用中更为复杂的过程都是基于以上的基本状态的转换。

2.2　生产者门户组件对象池算法

由于有多个消费者会注册到生产者, 因此生产者的远程门户组件 P 的实例对象数量可能非常庞大, 因此系统性能将成为关注的重点。我们使用门户组件对象池来管理运行时的远程门户组件对象。

传统的对象池是同一对象的多个可重用实例, 而生产者的组件对象池管理的是不同对象的多个实例, 因此就具有更大的复杂性。我们为每个门户组件维持一定数量的对象集合, 因此组件对象池设定了三个可配置参数以根据实际情况调整性能。这三个参数是对象总数量上限 Np, 单个门户组件对象数量初始值 Na 和对象缓存时间 Tc。

假设引擎可用内存资源为 Ma, 单个组件对象最多消耗内存 Ms, 可知道 Np≥Ma/Ms。我们提供一个测试函数, 可以方便地向管理员给出 Np 的参考值。

对象池运行一个监视线程, 每个未被引用的对象成为非活动对象, 每个非活动对象未被使用的时间如果超过 Tc, 就被销毁, 以保证系统资源的回收。具体算法如下:

(1) 初始化算法

对于每个门户组件, 预先生成 Na 个实例。

```
for( int i = 0; i < Na; i ++ )
{
new Portlet;  //生成一个组件对象
Tsi = setSpareTime;  //设置空闲时间
addPortlet;  //添加到实体池
}
```

(2) 对象监视算法

```
for( ; ; )
{
sleep;  //睡眠一定时间
Tn =  getCurrentTime( ) ;
for( allportlet)  //对每个组件循环
{
if( Tn -  Tsi > Tc)  //如果非活动时间大于对象缓存时间
removeportlet;  //从对象池中销毁组件, 释放资源
} }
```

(3) 对象获取算法

```
if( hasNotFreePortlet)  //如果没有空闲对象
{
if( n > Np)  //对象数量超出
release( portlet)  //释放一个未引用对象
newportlet;  //新建一个对象
addPortlet;  //添加到实体池
}
getPortlet( )  //获取对象
setUse( true)  //设置对象引用标志
```

测试结果表明, 使用复杂门户组件作为测试用例, 在未使用对象池的时候, 生产者每收到一个请求就新创建一个组件实例对象, 当请求数超出一定限度, 系统的内存资源就会很快耗尽, 抛出 OutOfMemory 异常。而使用了门户组件对象池, 避免了直接生成对象, 严格地将对象数量控制在限定范围 Np 内。以这种方式重新使用对象, 可在性能和扩展性方面显著获益, 避免了对象的重复创建和内存回收, 极大地提高了效率。

2.3　远程门户组件本地映射原理

生产者可以提供远程门户组件的实例对象给不同的消费者, 消费者可以通过生产者 Web 服务接口访问远程门户组件。通常情况下, 对于门户消费者, 使用的都是本地的门户组件, 每个门户组件需要实现特定的接口被门户引擎调用。使用 Ogc 表示本地组件的接口, 该接口返回一个 HTML 标记片段, 也即该组件的输出。每个门户页面表现为多个门户组件的输出的集合, 由门户引擎将本地的门户组件输出的 HTML 标记片段封装成最终用户页面。

为了使得门户可以使用远程组件, 我们使用一种特殊的本地组件, 这种组件实现了 Ogc 本地接口, 但是其返回的内容是在该组件内部调用了 Ogm 而获取到的。对消费者而言, 每个生产者对应一个 Web 服务的 URL, 每个生产者可能提供多个门户组件, 每个门户组件实例对象由一个唯一的句柄 PortletHandle 表示, 而且会在 Portal 端存在一个本地组件 Pyi, 因为有 Ogc(Pyi) = Ogm(URL, PortletHandle) , 也就能保证门户对远程组件的消费。

为了使消费者的工作具有重用性, 我们将所有调用远程门户组件的相关工作抽象为一个通用的消费者代理, 以功能组件的形式提供给门户。一个具体的映射过程如下:

(1) 门户站点管理员决定使用某个远程门户组件。

(2) 发送复制实例的 SOAP 调用请求, 生产者为门户生成新的门户组件实例。

(3) 门户生成一个本地门户组件与该远程组件相关联。

(4) 用户访问门户中的映射组件, 该组件调用远程组件的 Ogm 操作获取远程组件的输出, 通过 Ogc 接口返回给本地门户组件。

(5) 用户提交表单数据, 本地组件会转交给远程组件并将结果返回, 完全模拟远程组件的行为。

这样做的优点是, WSRP 门户组件对门户引擎透明, 具有极好的重用性、灵活性和扩展性。门户组件使用的组件都是本地的, 不同的是 WSRP 门户组件相当于远程门户组件在本地的一个映射, 这样所有与远程组件相关的操作都限定在组件内部, 对门户系统结构和性能没有任何影响, 具有很好的松耦合性。因为远程门户组件返回的是纯文本形式的 HTML 片断, 远程组件中输出的其他一些非文本信息将难以获取, 因此消费者在生成本地映射的组件同时, 将远程组件的二进制资源如图片、音频或视频下载到本地, 并且修改返回的 HTML 信息以保证页面中的资源完整地显示给最终用户。

3 系统实现

整个平台包括生产者和消费者两个部分组成。生产者表示为实现约定的 Web 服务接口集合和一定数量的门户组件的集合,因此生产者的设计实现就围绕着这两个核心。我们将 Web 服务接口单独实现为接口层,将门户组件抽象为实体,由引擎处理层管理实体对象,服务调用层负责调用最终的门户组件。而消费者的形式可能多样化,我们将集中于现有的系统,采用门户形式构建消费者,并兼顾到其他消费者扩展的要求。

系统的设计思路遵循兼容和开放原则,在实现标准接口的同时,不依赖于具体的门户平台,提供对多种门户组件的支持,并且采用通用的消费者代理,可以扩展出门户以外的消费者,如扩展的 JSP 标签等,体现了高度的兼容性和可扩展性。系统总体结构如图 4 所示,其中,消费者基于一个 Servlet 容器,而生产者基于 MBeanServer,使用 JMX 技术实现 WSRP 引擎来处理远程门户组件的生产。

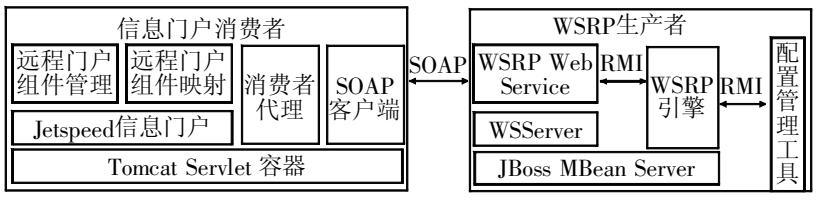

图 4 WSRP 运行管理系统结构图

(1) WSRP 生产者基础结构。设计实现一个独立于门户平台的生产者基础结构,实现规范中定义的 Web 服务接口;抽象出一个 WSRP 引擎来处理远程门户组件运行环境的共性需求,如会话处理、功能实现、持久化状态处理等。在引擎中使用抽象来的实体对象代表一个门户组件对象,以屏蔽不同门户组件之间的区别,进行统一管理;使用门户组件适配器接口,支持多门户组件实现,同时还要提供一个部署和配置工具。

整个生产者基础结构是一个 Web Service,目前采用本实验室自主开发的 Web Service 平台 WSServer[1],同时可以方便地迁移到如 Axis 等 Web Service 的 J2EE 实现。

(2) 信息门户消费者。我们给出了一个基于 Jetspeed 信息门户[8,9] 平台作为远程门户的消费者。在该信息门户中调用 WSRP 消费者代理消费生产者基础结构中的远程门户组件。

WSRP 消费者代理是我们设计的一个通用类库,该代理完成了大部分需要消费一个 WSRP 服务的共性处理,如会话、注册、描述、标记等以及 SOAP 客户端的工作。消费者代理对外提供一系列的功能接口,使得任何消费者都可以方便地通过消费者代理获取远程门户组件的标记。限于篇幅,消费者的具体结构不再详细探讨。

4 相关工作比较

Web 服务展现技术一直是各大研究机构和业界厂商的重要研发工作之一。目前基于 WSRP 实现的有 IBM WebSphere Portal Server,它集成了 WSRP 消费者和生产者的功能,是在自己的 PortletAPI 的基础上实现,基于 Servlet 接口,因此只能支持该平台上的远程门户组件[7];Apache 的 WSRP4J 项目所提出的 WSRP 服务器也是自支持、自定义的特定的远程门户组件接口,生产者是基于 Apache Axis 系统的 Web Service,结构比较简单,无法实现真正的跨平台组件重用。经过研究表明,WSRP4J 项目正在进行中,其实现上是针对每个请求都新生成一个门户组件对象,在性能上尚不能达到实用的程度。

本文通过研究远程门户组件的运行原理,给出 WSRP 平台的参考模型及其体系结构设计并实现了一个层次化的、统一的 WSRP 运行平台,为远程门户组件的共享和 Web 服务的展现提供了有力的支持。与现有的特定门户平台对 WSRP 的支持相比,本平台实现了专门针对远程门户组件的处理机制,在应用方面具有很强的灵活性。本系统的工作特点是:①生产者的前端是一个 Web Service 接口层,中间是一个 MBean 形式的 WSRP 引擎,其后端是服务调用层,这个三层结构体现了功能上的分离和松耦合的特点,使得系统具有良好的可移植性和可扩展性;②系统所有部分都采用模块化的功能组件设计,所有模块都由类工厂生成,类工厂在配置文件中是可配置的,使得每个模块成为一个封装完好,接口清晰的功能实现体,接口和实现相分离,具有良好的灵活性和易维护性;③抽象出通用的消费者代理,可以方便地扩展出如 JSP 标签消费者等其他消费者;④采用具有统一调用接口门户组件适配器的扩展机制,可以方便地支持多种类型的 PortletAPI。

5 结束语

目前 Web 服务技术的发展日新月异,WSRP 是一种新兴的 Web 服务规范,是 Web 服务走向应用的现实尝试。尽管目前还不是特别成熟,而且这些技术在目前来说很复杂,但是无疑在动态电子商务、ASP 应用等方面有着诱人的前景。随着硬件性能和网络带宽的发展,新的需求和新的应用模式层出不穷,WSRP 由于其面向用户、可交互、即插即用等优点一定会得到成熟的发展和长足的应用。总体说来,我们的系统基本实现了 WSRP 规范中所定义的接口和功能。在 Web 服务这个充满活力的领域,我们还有更多的工作要做,下一步我们将在运行效率、系统可靠性和服务质量保证等多方面开展研发工作,并通过应用实践来完善平台的设计与实现工作。

参考文献:

[1] 葛声,胡春明,等. 面向 Web Service 中间件的应用支撑环境[C]. 2002 '全国软件与应用学术会议(NASAC)论文集,2002.97-103.

[2] homas Schaeck. Web Services for Remote Portals (WSRP) Note 21 [EB/OL]. http://www-900.ibm.com/developerWorks/cn/webservices/ws-wsrp/index_eng.shtml, 2002.

[3] ilon Reshef. Building Interactive Web Services with WSIA & WSRP [J/OL]. WSJ Feature, 2002, (12):2-12.

[4] hristian Wege. Portal Server Technology[J]. IEEE Internet Computing, 2002, (6).

[5] homas Schaeck. Web Services for Remote Portlets White Paper[EB/OL]. http://www.oasis-open.org/committees/wsrp, 2002.

[6] lan Kropp, et al. Web Services for Remote Portlets Specification 1.0 [EB/OL]. http://www.oasis-open.org/committees/wsrp, 2003.

[7] homas Schaeck. WebSphere Portal Server and Web Services. [EB/OL]. http://www-3.ibm.com/software/solutions/webservices/pdf/WPS.pdf, 2002.

[8] d Anuff, et al. Web Services User Interface(WSUI) 1.0 Working Draft[EB/OL]. http://www.wsui.org/doc/20011031/WD-wsui-20011031.html, 2001-10-31.

[9] akeshi Chusho. Katsuya Fujiwara. A Form-based Approach for Web Services by Enduser-Initiative Application Development[J]. IEEE Conputer Society, 2002, (2).

作者简介:

赵勇(1979-),男,硕士研究生,主要研究方向为电子商务与信息门户;马殿富(1960-),男,教授,博士生导师,主要研究领域包括网络计算、科学计算可视化、海量信息处理。