

一种面向 Web 服务的门户组件机制的设计与实现

孙 璐, 葛 声, 马殿富

(北京航空航天大学 计算机学院, 北京 100083)

摘 要:通过门户组件(Portlet)来访问以 Web 服务形式提供的软件服务并展现其结果是通过门户(Portal)访问 Web 服务的一个核心问题。文中提出一种面向 Web 服务展现的 Portlet(以下简称 WSPortlet)机制来解决上述问题,通过遵循业界主流的 Portlet 规范,使用 XSLT 来对 XML 文档进行转换等途径保证了 WSPortlet 的规范性、有效性和灵活性,并且自主研发实现了 WSPortlet 系统(以下简称 WSPS)。

关键词:Web 服务; Portal; Portlet; WSPortlet

中图分类号:TP311.1

文献标识码:A

文章编号:1673-629X(2006)08-0010-03

Design and Implementation of Web Service - Oriented Portlet Mechanism

SUN Lu, GE Sheng, MA Dian-fu

(School of Computer Science and Engineering, Beihang University, Beijing 100083, China)

Abstract: It is a core problem of accessing Web service with portal to invoke a Web service - based software service and display its result. Provides a Web service presentation - oriented mechanism called Web service portlet mechanism to settle the problem. There are some approaches to ensure the normalization, validity and flexibility of this mechanism, such as obeying the chief Portlet specification in the field, flexibly using XSLT to transform XML document and so on. Finally, has achieved the above mechanism in the Web service portlet system which is researched and developed on one's own master.

Key words: Web service; portal; portlet; WSPortlet

0 引 言

目前互联网应用技术已经从简单的信息浏览发展到复杂的分布式应用。Web 服务是分布式应用的有效手段之一。它与传统的分布式技术相比具有更好的封装性、可集成性、开放性和互操作性,具有广泛的应用前景。但 Web 服务本身不包含展现样式和人机交互界面,这成为其应用普及的难点之一。而在互联网环境下,以 Portal 为代表的信息门户技术^[1]已经成为分布式应用主要展现手段之一。Portal 面向用户提供多种展现方式和统一访问入口,便于集成互联网上分散的数据服务或应用服务,同时提供个性化定制和单点登录机制^[2]。Portal 作为一个内容聚集的平台来访问 Web 服务成为上述背景下的一种重要工作模式。

Portal 通过组装一系列可插式页面组件 Portlet 来实

现信息的展现。Web 服务一般采用 XML 为数据格式,通常其输入输出都不包含展现内容,需要在 Portal 中提供方法来访问和展现 Web 服务^[3]。目前通过 Portal 来访问和展现 Web 服务的工作方式主要有两种:一种较简单的方式就是在作为 Portal 组件的 Portlet 中针对各个 Web 服务进行特定编码来访问和展现,这种方式在实际应用中会带来开发周期长、重复编码多、数据和展现混杂、Portal 和 Web 服务耦合过于紧密等问题^[3];另一种方式是结构化信息标准促进组织(OASIS)颁布的 WSRP 规范^[4]所提出的展现方式,其原理是把 Portlet 进行 Web 服务化,将 Portlet 作为 Web 服务处理结果返回给 Portal,WSRP 增加了 Web 服务端的工作量和调用过程中的数据量,同时需要 Portal 和 Web 服务都必须支持 WSRP 规范,制约了它的发展和应用。

针对上述问题,文中设计开发了一种面向 Web 服务展现的 WSPortlet 机制,它提出一种有效的面向 Web 服务展现的 Portlet 规范,并构造其运行支撑环境 WSPS,在一定程度上克服了现有工作中的上述种种不足。文中主要完成了以下工作:定义了一类符合业界 Portlet 规范的 WSPortlet 的结构组成,主要包括 WSPortlet 及其相关 Web 服务的结构、内容和开发规则;设计了 WSPortlet 面

收稿日期:2006-01-20

基金项目:国家“八六三”计划基金资助项目(2003AA115420;2004AA112030;2004AA115110)

作者简介:孙 璐(1981-),男,安徽巢湖人,硕士研究生,研究方向为电子商务与中间件技术;马殿富,博士,教授,博士生导师,研究方向为计算机理论、分布式计算、网络计算、中间件技术。

向 Web 服务展现的调用机制,完成了对 Web 服务的访问及其输入输出的界面处理;实现了对 WSPortlet 灵活高效的自动化处理等内容。

1 相关规范

目前,业界已经形成了面向 Portlet 研发和 XML 转换的规范,为保证 WSPortlet 的规范、有效,其设计实现必须依据以下技术规范:

1)JSR168 Portlet 规范。

Java 社区过程(JCP)于 2003 年 10 月颁布了 Java 规范请求(JSR)168 号:Portlet Specification 1.0^[5] 版本。此规范提出了 Portal 基本体系架构和 Portlet 接口的标准,业已成为业界主流的 Portlet 规范。WSPortlet 的组成定义完全遵循此规范提出的接口标准。

2)XSLT 规范。

万维网联盟(W3C)于 1999 年 11 月颁布了可扩展样式表转换语言(Extensible Stylesheet Language Transformations, XSLT)1.0 版本。XSLT 是把 XML 文档转化为另一类格式文档的 XML 转换语言规范,其可以最大限度地数据、编辑和表现相分离,具有良好的可维护性,易于应用到多种客户设备和语言。在 WSPS 中, WSPortlet 使用 XSLT 处理器所提供的转换功能,依据从 Web 服务处得到的此服务的 XSLT 样式表,将不包括展现内容的 XML 格式的 Web 服务数据转换成 HTML 格式的展现页面。

2 设计实现

文中所提出的 WSPortlet 机制和 WSPS 系统包括了 WSPortlet 的组成定义、对 WSPortlet 进行自动化处理的模块和对 Web 服务进行 WSPortlet 化改造等内容。文中设计实现的 WSPS 系统包括 Portal 子系统和 Web 服务子系统两部分:Portal 关注于 WSPortlet 及其生命周期(生成、部署、运行和维护)的自动化处理;Web 服务关注于可被 WSPortlet 作为客户端来调用的特定类型 Web 服务;它们之间的交互数据采用 XML 格式。WSPS 系统框架如图 1 所示。

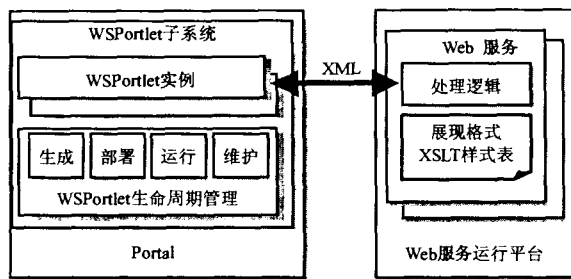


图 1 WSPortlet 系统框架

2.1 WSPortlet 组成定义

WSPortlet 作为一类 Portlet,遵循业界相应的 Portlet 规范,实现了 JSR168 所提出的 Portlet 接口。文中设计的 WSPortlet 遵循 MVC 设计模式。其组成定义及模块间关

系如图 2 所示。

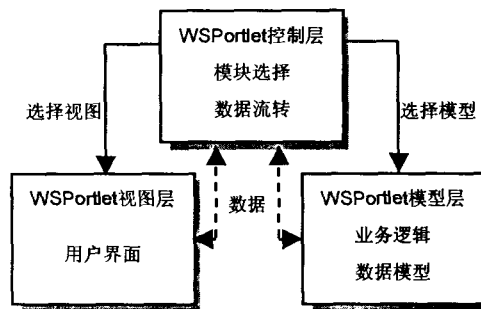


图 2 WSPortlet 的组成定义

WSPortlet 模型层是 WSPortlet 的核心部分,在其中实现了 WSPortlet 的业务逻辑和数据模型。业务逻辑就是调用与此 WSPortlet 关联的 Web 服务的过程,表现为一系列实现类;数据模型就是以 XML 文档形式存在的 Web 服务数据以及作为这些数据的展现格式的 XSLT 样式表。业务逻辑和数据模型构成了 WSPortlet 对 Web 服务的调用机制,这也是 WSPortlet 机制研发的重点和难点。WSPortlet 调用机制的核心算法如下所示:

```
//JSR168 规范规定的 Portlet 操作接口
//从 Portlet 容器接收 ActionRequest
//向 Portlet 容器返回 ActionResponse
processAction (ActionRequest, ActionResponse) {
1) getReqData (ActionRequest);
   //从 ActionRequest 中提取请求数据 ReqData
2) getPortletconf (ActionRequest);
   //从 ActionRequest 中提取 WSPortlet 注册信息
3) if (ReqData != NULL)
   createXML (ReqData);
   //将请求数据封装成 XML 文件
4) DataHandler (XMLReq);
   //构造 SOAP 报文,将上述 XML 文件作为附件
5) if (portletconf != NULL)
   getWSInfo (portletconf);
   //从注册信息中提取 Web 服务信息;
6) if (Web 服务可访问)
   callWS (ReqSOAPMes, ResSOAPMes);
   //调用 Web 服务;
7) if (SoapMessageRes != NULL)
   resolveXML (ResSOAPMes);
   //解析返回的 SOAP 附件,得到 XML 文件
8) getXSLT (portletconf);
   //由 WSPortlet 注册信息得到 XSLT 样式表文件
9) XSLT.Xalan (XMLRes, XSLT);
   //调用内嵌的 XSLT 处理器,生成 Portlet 页面
10) putResData (PortletView, ActionResponse)
   //将 Portlet 页面放入返回消息 ActionResponse 中
}
```

WSPortlet 视图层将 Web 服务的输入页面展现给 Portal 用户,由用户填写并且提交请求数据,以及将 WSPortlet 模型层返回的 Portlet 页面作为输出页面,供用户

察看 Web 服务的处理结果。

WSPortlet 控制层控制 WSPortlet 各个运行步骤的次序以及数据在各个层次之间的流转。文中工作参考了 apache 基金会的开源 Portal 项目 Jetspeed 所提供的对 Portlet 进行引用的 Portlet 桥接技术,在现有 Portlet 桥接器的基础上扩展了一类 WSPortlet 桥接器,来在 Portal 中对 WSPortlet 进行连接、访问和控制。WSPS 在此层面上实现了对 JSR168 规范^[5]中要求的 Portlet 模式和窗口状态的支持以及两者组合情况下的 Portlet 操作权限控制。WSPortlet 控制层还完成了不同角色用户对不同模式和状态下的 WSPortlet 的使用权限控制。

通过实现以上三个层次,完整地开发了一个 WSPortlet,可以在其中有效地调用相应的 Web 服务并且展现其处理结果。

2.2 WSPortlet 生命周期管理

WSPS 能够自动灵活地处理 WSPortlet 生命周期中的生成、部署、运行和维护等各个阶段工作。生成模块根据用户输入信息生成一个 WSPortlet 部署单元,并且相应地提供部署单元的销毁功能;部署模块将一个部署单元所定义的 WSPortlet 在 Portal 中进行发布和注册,使 Portal 用户可以订制和使用此 WSPortlet,部署模块也相应地提供取消部署的功能;运行模块和 WSPortlet 的控制模块相结合,为各个 WSPortlet 运行实例提供运行时的支持和管理,主要实现了对运行时的基本支撑功能和对一些运行时数据的日志功能;维护模块可以对 WSPortlet 的注册信息进行修改,以及当相应的资源发生更改时,对此 WSPortlet 重新进行自动化的开发和部署,将依据新资源重生成的 WSPortlet 提供给用户使用。生命周期管理各个模块的划分和概略的工作过程如图 3 所示。

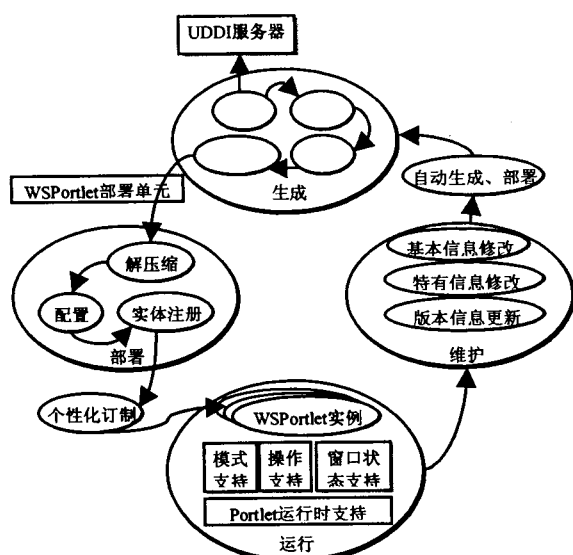


图 3 WSPortlet 生命周期

以上就是 WSPS 对 WSPortlet 的生命周期的管理,贯穿了一个 WSPortlet 的生成、部署、运行和维护等阶段。上述这些过程也都以 Portlet 的形式提供给 Portlet 开发者

使用,其内部工作过程对于 Portlet 开发者而言都是透明的。Portlet 开发者只需要填写和选择一些基本的配置信息,WSPS 就可以自动地完成对 WSPortlet 的开发、管理和维护。

2.3 Web 服务的 WSPortlet 化

WSPortlet 是面向 Web 服务的一类 Portlet,WSPS 对 WSPortlet 所对应的 Web 服务的资源中所包含的内容作了一些的扩展,增加了此 Web 服务的处理数据的展现格式文档。

WSPS 向 Web 服务开发者提供将 Web 服务进行 WSPortlet 化的模板,此模板以类包的形式存在,主要提供给 Web 服务客户端获取此 Web 服务的展现格式文档(XSLT 样式表文件)及其版本信息(时间戳、版本号等)的方法。Web 服务开发者在开发 Web 服务时使用这些方法即可,无需额外编码。

3 结束语

综合来看,目前在分布式环境下存在着在 Portal 中访问和展现 Web 服务的问题,主流的 Portal 开源项目诸如 Liferay Portal,apache Jetspeed 和 JA-SIG uPortal 等都尚不能够提供相应的处理机制来解决此问题,主流的 Portal 产品诸如 BEA Weblogic Portal 和 IBM Websphere Portal 等目前也都尚未明确提出对此问题的解决办法。

文中工作针对上述问题研究设计了面向 Web 服务展现的 Portlet 机制,自主实现了 WSPortlet 系统,其工作具有如下特点:

- ①研发的 WSPortlet 完全符合业界主流 Portlet 规范;
- ②WSPortlet 调用机制遵循 XML 文档转换规范,能够便捷的、透明的在 Portal 中调用和展现 Web 服务;
- ③研发的 WSPortlet 采用 MVC 设计模式,具有很好的强壮性、弹性和易于维护性;
- ④WSPS 对 WSPortlet 提供全面的支持功能,便于对其灵活高效地开发、管理和维护;
- ⑤通过使用 Web 服务的 WSPortlet 化开发模板,能够快捷地开发符合 WSPortlet 要求的 Web 服务。将来,将在应用实践中进一步完善 WSPortlet 机制和 WSPS 系统,并进行大规模并发访问下的性能和可靠性评测等工作。

参考文献:

- [1] Wege C. Portal Server Technology[J]. IEEE INTERNET COMPUTING,2002,6(3):73-77.
- [2] Bellas F. Standards for Second-Generation Portals[J]. IEEE INTERNET COMPUTING,2004,8(2):54-60.
- [3] 葛声,马殿富,胡春明.基于 Web 服务的网络软件运行平台研究与实现[J].北京航空航天大学学报,2003,29(10):897-900.
- [4] Allamaraju S,Brooks R. Web Services for Remote Portlets 1.0 Primer Committee Draft 1.00[EB/OL]. http://www.oasis

(下转第 15 页)

问控制服务器:Telnetd-0.17;电子邮件服务器:Qmail-1.03;以太网配置 100M。

为验证原型系统实现的 NAT-PT 过渡机制能否很好地满足实际应用的需求,对原型系统中 NAT-PT 转换网关的基本功能和性能以及 NAT-PT 和各个 ALG 协同工作的性能进行了测试。

3.1 整体传输性能及分析

首先,经过对原型系统的调试,成功实现了基于 NAT-PT 转换网关的多种基本服务(WWW,DNS,E-mail,Telnet 和 FTP)的 IPv4/IPv6 间的过渡应用,功能正常,性能稳定。

其次,测试比较在纯以太网环境下以及在经过 NAT-PT 转换的情况下,ICMP 分组的往返时间 RTT(round-trip time)、TCP 连接建立的响应时间和 FTP 传输 400MB 文件的速度,结果记录在表 1 中。

表 1 不同协议环境下传输性能比较

	纯 IPv4 环境	纯 IPv6 环境	IPv4 to IPv6	IPv6 to IPv4
ICMP 往返时间(ms)	0.160	0.143	0.516	0.524
TCP 连接延时(ms)	4.933	4.906	5.434	5.532
FTP 传输速度(MB/s)	9.6	11.0	11.0	10.0

分析表 1 中的数据可以发现:

①整体上,NAT-PT 过渡环境下的各项指标均接近于纯以太网环境。这表明该过渡机制对 FTP 的传输性能影响不大,系统整体性能良好。

②NAT-PT 过渡环境下的往返时间、连接延时比纯以太网环境下长,传输速度比纯以太网环境下慢。这主要有两个方面的原因:第一,NAT-PT 本身对分组的转换过程需要一定的时间;第二,NAT-PT 是网络层软件,FTP-ALG 更是处在应用层,每个分组的传输都要经过转换网关服务器从底层到高层的多层处理,这也造成时间上面的损耗。

③基于 NAT-PT 技术的双向连接性能存在一些差异,IPv6-to-IPv4 情况下 FTP 建立连接响应时间和文件传输速度稍慢于 IPv4-to-IPv6。这种性能上的差异是由混合型 NAT-PT 内部地址映射机制的不同造成的。在 IPv6-to-IPv4 情况下采用的是地址端口映射机制 NAPT,它是由一个 IPv4 地址通过不同的端口号映射到不同的 IPv6 地址,分组在通过 NAPT-PT 转换时,除了在 IPv4 地址池的地址查找外,还要对同一个地址的大量端口号进行查询才能找到正确的映射地址,从而造成了 IPv6-to-IPv4 的性能下降。

3.2 FTP 传输性能分析

鉴于 FTP 服务对系统整体性能全面性的反映,单独对 FTP 进行服务性能和 FTP-ALG 处理能力的测试:取

100 个大小在 0~700MB 之间的文件,分别测试它们在 IPv4/IPv6 间经过 NAT-PT 转换的传输速度,记录测试数据并绘制关系曲线图,如图 3 所示。

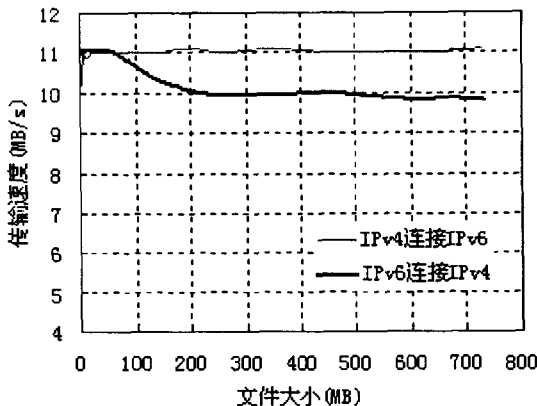


图 3 FTP 传输速度测试

分析图 3 可以发现,无论文件大小怎样变化,经由 NAT-PT 网关转换的传输速度非常稳定,在 NAT 机制下基本保持在 11MB/s,NAPT 机制下基本保持在 10MB/s,这样的传输性能已可以满足各种多媒体应用的实际需要。另一方面,从两种不同机制下传输速度的差异可以看出,NAPT 机制虽然更加有效地利用了 IPv4 地址资源,但是性能上受到了一定的影响。

4 总 结

文中研究和实现的 NAT-PT 转换网关已被应用于中日下一代互联网 IPv6 合作项目中。在实验环境下,基于该系统来实现的 IPv4/IPv6 基本服务的过渡应用功能完善、性能良好,具有很高的可用性和实用性。下一步工作是将把 NAT-PT 过渡技术实际部署到校园网内,并通过 CERNET 华东北网络中心接入 CERNET 骨干,在实际环境中进一步测试和改善 NAT-PT 的功能和性能。

参考文献:

[1] IETF RFC 1883. Internet Protocol, Version 6 (IPv6) Specification[S]. 1995.
[2] IETF RFC 2893. Transition Mechanisms for IPv6 Hosts and Routers[S]. 2000.
[3] IETF RFC 2766. Network Address Translation - Protocol Translation (NAT-PT)[S]. 2000.
[4] IETF RFC 2663. IP Network Address Translator (NAT) Terminology and Considerations[S]. 1999.
[5] IETF RFC 2765. Stateless IP/ICMP Translator(SIIT)[S]. 2000.
[6] IETF RFC 2428. FTP Extensions for IPv6 and NATs[S]. 1998.

(上接第 12 页)

- open.org/committees/download.php/10539/wsrp-primer-1.0.html,2004-12-03.
[5] Abdelnur A (Sun Microsystems Inc), Chien E(Sun Microsys-

tems Inc), Hepper S (IBM). JSR 168: Portlet Specification [EB/OL]. http://www.jcp.org/en/jsr/detail?id=168, 2003-10.