Web Services 技术应用与探讨

胡方霞[1] 曾 一[2] 高 旻[2]

(重庆广播电视大学理工学院 重庆 400052)[1] (重庆大学计算机学院 重庆 400030)[2]

摘 要 Web Services 是一种使程序的功能通过网络与其他服务交互的一种机制。文中,分析了何时利用 Web Services、Web Services 在系统中扮演的角色、以及 Web Services 设计注意事项,并通过一个网络学院的实例对这些内容进行了详细的探讨。

关键词 Web Services,XML,SOAP,体系结构

The Research and Application of Web Service Technology

HU Fang-Xia[1] ZENG Yi[2] GAO Min[2]

(Chongqing Radio & Television University Engineering College,Chongqing 400052)[1]

(College of Computer Science,Chongqing University,Chongqing 400030)[2]

Abstract Web services are a mechanism for exposing program functionality over the Web, typically to other services. In this paper we analyze when to utilize Web Services, the role that Web Services acts in the system and the precautions in Web Services design, at last we have carried on the detailed discussion to these contents through the instance of a network institute.

Keywords Web Services,XML,SOAP, Architecture

1 引言

随着分布式计算、网络以及面向对象技术的发展,传统的C/S模式已经不能满足开发者的需求,分布式对象技术迅速发展起来。与传统的C/S模式相比,分布式对象技术通常表现为n层结构,并且能够通过中间件机制来屏蔽硬件平台的差异性和操作系统与网络协议的异构性,从而适应异构的网络计算环境。主流的分布式对象模型有微软的COM/DCOM和.NET策略、Sun公司的Java/RMI和OMG的CORBA,这些技术使企业内部的分布式应用的开发变的相对容易[1]。但是,这些模型构造的系统要求客户端必须使用特定的协议访问服务器端的对象,也就是说服务客户端和系统提供的服务本事必须是紧密耦合的,要求一个同类基本结构。如果要在不同的模型构造的体系结构之间提供互操作性的话,需要建立特殊的桥梁,交流将变的非常复杂。因此需要提供一个开放式架构(独立于任何语言和体系结构),此架构可以被任何平台访问,不管他是服务器、桌上电脑还是移动设备。Web Services正是迎合了这种需要而发展起来的,它是松散耦合的,并使用了标准的Internet协议HTTP、XML等[2]。

XML Web Services是构件服务的通用网络接口。为了支持所有体系结构之间的互操作性,XML Web Services使用了W3C协议,例如XML、WSDL、SOAP和UDDI协议。这使得各个公司可以通过网络互相使用相似的服务、开发过程和工作模式。XML Web Services可以被看作一个特殊的程序与程序之间的表示层,在理想状况下,该层能够被组织、维护更新而不必重新设计底层的软件。

本文第2节将简明的描述Web Services及相关标准与协议;第3节将深入研究Web Services的应用和Web Services的角色模型;第4节描述一个包含基本组成的Web Services设计与调用的实例。

2 Web Service的发展与底层协议

2.1 Web Service的发展

流行的面向对象中封装和信息隐藏的原则在构件模型中已经得到了长足的发展(例如COM和CORBA构件),它们的执行语句包含在用IDL、MIDL等独立语言编写的制定接口中。目前多数构件模型是基于特殊标准的,如果要进行跨平台的互操作通常需要在构件框架间建立通信软件[1]。

因此,有必要定义一个开放的、分布式的、程序间通信的模型。IBM公司首先引入了XML Web Services的概念:XML Web Services是自包含的、模块化的应用,它能够在网上被描述、发布、定位和激活。开放通用的协议标准使它对所有的平台和体系结构都是可行的。

2.2 底层协议

2.2.1 XML语言

XML是语言定义语言,是SGML的一个子集。XML的特征如下:1)XML是一种元语言,允许用户创建自己的标记语言;2)XML是一个开放标准,文档与具体的平台和语言无关;3)XML将数据和对数据的处理分开;4)XML能精确定义数据,不同结构的XML文档可以通过XSLT互相转换。Web Services的各个协议是基于XML的。

2.2.2 SOAP协议

SOAP是Web Services技术的核心之一,它提供了一种可通过多种底层协议进行交换的消息框架。它是一种轻量级协议,用于在分散型、分布式环境中交换结构化信息。SOAP协议包括4个主要部分:SOAP封装、SOAP编码规则、SOAP-RPC表示和SOAP绑定。

1)SOAP封装定义了一个框架来描述消息的内容是什么,是谁发送的,谁应当接收并处理它以及如何处理。SOAP编码规则。

胡方霞 讲师,研究方向为数据库技术;曾 一 教授,硕士生导师,研究方向为软件工程。

2)SOAP 编码规则是可选项,用来表示应用程序的数据类型的实例。

3)SOAP-RPC 表示定义了一个请求/响应消息交换系统。

4)SOAP 绑定是传送 SOAP 封装的一种方法,它详细说明了 SOAP 和 HTTP 之间的绑定。SOAP 绑定是可选项,可以用其他的传输协议例如 SMTP,FTP 来传送 SOAP 封装。

SOAP 的一个主要优点就是简单,SOAP 使用 HTTP 作为网络通讯协议,接受和传送数据参数时采用 XML 作为数据格式,即使用 HTTP 传送 XML。HTTP 并不是高效的通讯协议,XML 还需要额外的文件解析,这使得交易的速度大大低于其他方案,但由于 XML 是一个开发、健全、有语义的信息机制,HTTP 又能避免防火墙问题,所以 SOAP 得到了广泛的应用。

2.2.3 Web 服务描述语言 WSDL

WSDL 是一种 XML 文档语法,它为 Web Services 提供了一个公共的约定,支持 Web Services 的发布和发现。WSDL 被定义为一个"XML 格式的,描述网络服务消息的一系列端点操作"。

WSDL 描述说明的是 Web 服务的三个基本属性:1)服务做些什么-服务所提供的操作与方法;2)如何访问服务-数据的格式以及访问服务操作的必要协议;3)服务位于何处-由特定协议决定的网络地址,如 URL。

2.2.4 统一描述、发现和集成协议 UDDI

一旦部署了 Web 服务,潜在用户就必须能够发现它在什么地方以及它如何工作。UDDI 是一种行内规范,它定义了一项基于 SOAP 的协议,用于更新和查询 Web 服务信息库。UDDI 可以发布并发现 Web 服务,最大限度地访问站点并获得最终的成功。

3 Web Services 应用和角色模型

Web Service 具有跨平台的优点。但是否所有的应用程序都应该应用 Web Service 呢? 如果应用 Web Service,从体系结构的观点看,它和应用程序的关系是什么,应该注意什么呢?

3.1 何时应用 Web Services

Web Services 是一个基于 XML 的标准,它可以被任何平台、任何语言编写的程序访问。当一个方法需要供大量不同平台或不同编程语言调用时,就应该考虑用 Web Services 了。举个例子:假设有一个天气预报的功能模块,接收的参数是日期和城市名,返回一个关于天气的字符串。此模块可能有大量的应用程序调用,比如农业网站、手机服务程序、QQ,模块的供应商不能预知调用此模块的应用平台,也不能预知调用此模块的应用程序是用什么语言编写的,是 Java, C#,还是 VB, Delphi 或是 flash? 此时就应该应用 Web Service。

然而,应用了 Web Service 的程序,在通过 SOAP 构造时,大量 XML 分解需要串行化类型系统和串行化要调用的方法,所以,调用 Web Services 时需要足够的带宽,而且 Web Services 的执行效率却不如基于 CORBA、EJB 和 DCOM 平台的二进制系统。因此,当系统需要良好的性能且不需要于与不同的硬件或操作系统的客户进行交流时,二进制系统应优于 Web Services。

3.2 Web Services 在系统中扮演的角色

正如本文第 2 部分所解释的,Web Services 通过提供 WSDL 约定生成通用接口。从体系结构的观点来看,它形成了程序与程序之间的表示层,这一层是在商业逻辑和应用层之外的。将这一层称为 Web Services 层[3]。

Web Services 层应该描述多少细节和过程? 开发者应如何看待把现有的系统模块转换成 Web Services? 现在流行的系统有三层结构:数据库、商业逻辑层和表示层,这种结构实现了数据、商业逻辑和用户接口的分离,各层的子系统都可以在不影响其他层或子系统的前提下被重用、扩展或修改。三层系统中的每一层都在系统的生命周期中扮演不同的角色,当 Web Services 被引入时,它究竟扮演什么样的角色? 本文的观点是,开发者应该编写接口而不是编写执行程序。也就是说 Web Services 应该仅仅是向用户发送信息和为底层系统接收信息。

因此,当 Web Services 被引入到系统时,应该简单地形成另外一层,即网络接口层,此层与用户接口层扮演类似的角色,只不过用户接口层提供的是用户与系统交互的接口,而 Web Services 层提供的是用户程序与系统交互的接口。图 1 描述了 Web Services 与现在流行的三层结构的关系。

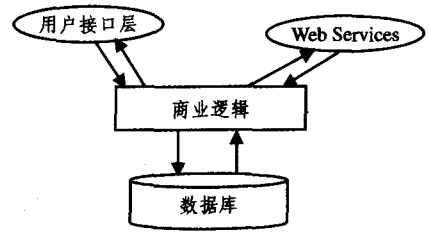

图 1 Web Services 层与现在流行的三层结构的关系

3.3 Web Services 设计注意事项

把一个系统模块公开为 Web Service 为系统设计引入了新的问题。开发者怎样才能最好地设计 Web Service 层,使其他服务与特定的 Web Service 进行安全的交互呢?

首先,不管信息是来还是去,都不需要在网络交互模型中保持持久的连接。因此,当带宽和效率成为问题时,需要最小化远程事务。

其次,Web Services 是程序与程序间的通信,某些进程可能需要在调用其他进程之前正确完成,这就需要有一个排队机制。例如,在取款操作前需要先要得到资金可用的正确响应。

最后,因为 Web Services 在开放的网络协议中执行,必须执行安全保护模型以保证信息的安全性与无害性,因此应该在恰当位置对要传输的数据进行加密,并且用户需要有密码和特定的安全属性才能执行受保护的过程。

4 Web Services 技术应用

4.1 应用实例

目前开发的网络教育学院的项目,由于总部开发的服务模块可能被各地学习中心编写的应用程序所调用,而这些应用程序所使用的平台和语言是未知的,因此需要应用 Web Services 技术。例如项目中"报名"功能模块,开发平台是微软的.NET(由 C#语言编写),将其暴露为 Web Services 后就可以供任何语言编写的应用程序所调用。项目提供了多种方法实现报名:1)Asp.net 开发的学生自主报名界面(应用 C#语言),通过调用功能业务模块实现;2)批量报名处理子系统(语言不限),这些子系统可以通过 Web Services 调用报名功能模块来实现。从这个实例来看,Web Services 在项目中只起到一个程序开发接口的作用,图 2 是它的应用实例图。

如何保证 3.3 小节提出的三点注意事项呢? 首先,不保持持久的连接。批量报名处理子系统是在各教学中心运行,

如果各教学中心都一直保持连接，每输入一个报名学生信息就上传至服务器，就会出现带宽和效率问题，因此应该尽量减少这种持久的连接。解决的方法是：各中心可以脱机输入多个学生信息，当希望上传时再进行连接。其次，需要排队机制。Web Services 方法的调用需要一定的顺序性，例如在进行报名时段前必须进行用户权限检查，在调用上传数据前必须保证时间是在报名时段内。为每个过程设定变量，并在调用各过程前判断前一个过程对应的变量值是否合法即可保证调用的顺序。最后是信息的安全性问题，利用 SOAP 头进行用户程序的身份验证，并且对用户的密码进行 MD5 加密，对关键过程只有数据库中合法的有权限的用户才能调用。

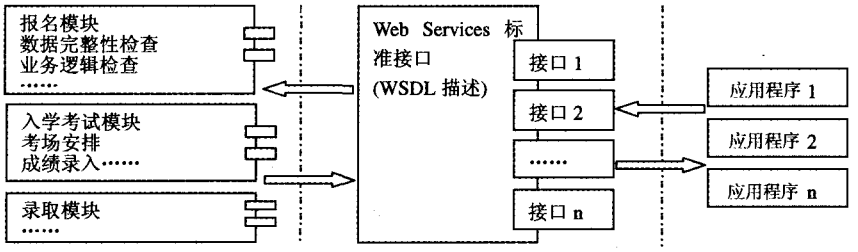

图 2　Web Services 应用示例图

4.2　接口的基本组成

每个 Web Services 都至少应该具有实现安全检测的接口（例如用户验证）和信息服务接口，然后根据业务逻辑的需要再添加其他的接口。

4.2.1　实现安全保障的接口

为了保证原系统的安全，需要定义一个安全接口。这个接口为认证提供了多个参数，这些参数中至少有两个字符串域：用户名和密码，当访问受保护数据服务中的方法时就需要这两项。还有一个保证安全的机制是把特定的方法做为临界方法：当一个方法被执行时，判断它的重要性，如果是重要方法就作为临界区，如果一个方法是临界的（例如判断用户合法性的方法），它的失败将导致整个事务的失败。

4.2.2　信息服务接口

在 Web Services 中应该有一个通用的信息服务接口，此接口可以被其他应用程序或其他服务调用，它仅仅是网络和商业逻辑之间传送数据的通道。此接口的基本的方法就是获取和发送数据。获取数据方法从商业逻辑提取数据并提供给调用者，发送数据方法是向商业逻辑提供必要的参数。通常来说，服务应该能够在不重新设计底层系统的情况下实现这些方法。

4.3　Web Services 实例代码

4.3.1　应用 SOAP 头实现安全保障

1）实现方法：首先生成一个自定义的 http 模块并在 Webconfig 中注册；在 Web Services 类中添加一个 WebServiceAuthenticationModule 定义的 SOAPAuthHeader 字段类型，并将 SOAPAuthHeader 属性应用于 Web Services 方法中，然后使用赋给这个字段类型的名称 sHeader 定义这个属性的构造函数；对于客户端来说，每一次将请求发送到 Web-Services 都需要填充 SOAP 头，在 WebServices 定义了 SOAP 头之后，客户端的代理类也会生成相应的类代理。

2）主要代码

```
public class AuthHeaderCS : SoapHeader  //由 SoapHeader 扩展而来的 AuthHeader 类
{public string Username;  public string Password;}
public AuthHeaderCS sHeader;
[SoapHeader("sHeader")]
public string SecureMethod()
{  if (sHeader==null) return"错误：请输入登录信息";  //如果 SOAP 头为空，提示错误
   string usr=sHeader.Username,pwd=sHeader.Password;
   if (AuthenticateUser(usr, pwd)) return"登录成功:";  //调用 AuthenticateUser 验证用户身份
   else return "错误：未能通过身份验证";
}
private bool AuthenticateUser(string username, string password)
{if ((username ! ="")&&(password ! =""))  //检验数据库中是否存在此用户
   {  EntityData entity = SystemUserAccountInfoManagement.FindByPrimaryKey(username);
      if(entity.Tables[0].Rows.Count==0)  return false;
      ……
   }
}
```

4.3.2　调用 Web Services 时对 SOAP 头的填充

1）实现方法：客户端每一次把请求发送到 WebServices 都需要填充 SOAP 头，首先创建一个服务的对象实例代表 Web Services，然后创建一个 SOAP 头对象实例，并填充 SOAP 头，接下来将 SOAP 头与该代理程序相关联，以便使用 Web Services。

2）主要代码

```
RecruitEnterService hRes=new RecruitEnterService();  //创建一个服务对象的实例 h
AuthHeaderCS myHeader=new AuthHeaderCS();
//给 myHeader 的成员变量赋值，Password 采用 MD5 加密
myHeader.Username=this.txtName.Text.Trim();
myHeader.Password=FormsAuthentication.HashPasswordForStoringInConfigFile(this.txtPwd.Text.Trim(),"MD5");
hRes.AuthHeaderCSValue=myHeader;
hRes.PreAuthenticate=true;
hRes.Credentials=System.Net.CredentialCache.DefaultCredentials;
if(hRes.SecureMethod().ToString()=="1")//检查身份验证通过
    if(hRes.IsRecruitTimeOut())//检查是否超过报名时段
    {……
    DataSet UREIMDS = hRes.UpLoadRecruitEnterInfoMethod(UpDS);
    ……}
```

结束语　Web Services 是构件服务的通用网络接口，可以支持所有应用程序之间的互操作。本文研究了支撑 Web Services 技术的 XML、WSDL、SOAP、UDDI 等开发的网络标准；分析了何时适用 Web Services、Web Services 在系统中扮演的角色以及 Web Services 设计注意事项；并通过一个网络学院的实例说明了为何使用 Web Services，Web Services 在项目中的角色等问题。目前基于 Web Services 的应用尚处于发展阶段，有待于进一步标准化，但由于其潜在着商业和技术价值，必定有非常广阔的发展前景。

参 考 文 献

1　吴敏强，张潇，杨书慧. 从分布式对象到 Web 服务. 计算机科学，2002，29（11）：12～16

2　郑小平. .NET 精髓 —Web 服务原理与开发. 人民邮电出版社，2002

3　Nioloudis N，Mingins C. XML Web Services Automation：A Software Engineering Approach. In：Proceedings of the Ninth A-sia-Pacific Software Engineering Conference（APSEC'02），IEEE 2002

4　Paul HollandBuilding. Web Services From Existing Application. eAI Journal，2002(9)