广西科学院学报
Journal of Guangxi Academy of Sciences

2009,25(4):308~310
Vol. 25,No. 4 November 2009

基于 Hessian 协议的轻量级 SOA 系统集成方案及其实现技术

A Lightweight Solution and Implementation Technique for SOA Application System Integration Based on Hessian Protocal

吴珊娜[1,2],覃 华[1]
WU Shan-na[1,2],QIN Hua[1]

(1.广西大学计算机与电子信息学院,广西南宁 530004;2.广西中医学院,广西南宁 530001)
(1. School of Computer, Electronics and Information, Guangxi University, Nanning, Guangxi, 530004, China; 2. Guangxi Traditional Chinese Medical University, Nanning, Guangxi, 530004, China)

摘要:使用 Hessian 协议实现轻量级面向服务架构(SOA)集成方案,并通过一个高校人力资源 SOA 系统整合案例介绍该方案的实施过程。该方案具有易学易用、数据交换效率高、过防火墙能力强等优点,具有较好的工程应用价值。

关键词:信息系统 SOA Hessian 协议 系统集成
中图法分类号:TP311 **文献标识码**:A **文章编号**:1002-7378(2009)04-0308-03

Abstract: A lightweight solution for service oriented architecture (SOA) integration based on hessian protpcal is proposed and the process of the implementation of this solution is shown with a case of university human resources SOA system integration. This solution is smiple, high efficiency in data transmission and strong capability in crossing the firewall. It is valuable in engineering application.

Key words: information system, SOA, Hessian protocal, system integration

企事业单位内部通常存在许多相互独立运行的信息系统,这些系统形成一个个信息孤岛。人们非常希望花费最小的代价,将各个信息孤岛整合起来,实现信息共享,提高资源的利用率。现在一般使用面向服务架构(SOA)的思想解决此类问题[1]。SOA 的基本理念是把信息系统中需要整合的业务通过服务和接口联系起来。接口是中立的,与开发平台和编程语言无关,这使得通过服务整合异构信息系统成为可能。由于 SOA 只是一个理念或概念模型,没有具体的实现技术方案,所以给 SOA 具体实施带来困难。目前研究较多的是基于 SOAP 协议 Web 服务的 SOA 解决方案。由于 SOAP 协议体系结构较为复杂,学习和推广难度较大;而且 SOAP 协议采用 XML 规范来交换数据[2],存在效率低、安全性不高等问题。

Hessian 协议[3]是一个轻量级的、自描述的二进制 RPC 协议,它主要用于面向对象的消息通信。Hessian 协议和 SOAP 协议都是为了实现异构系统间的数据交换。它们的相似点在于都是将协议报文封装在 HTTP 封包中,通过 HTTP 信道进行传输,因此可以不受防火墙的限制。Hessian 协议不是一个大型的框架,具有精简的特性,适合于传送二进制数据,同时并不需要继承任何相关的协议。相对于 SOAP 协议,Hessian 协议在编码方面具有很大的优势,它将本地格式的数据编码为二进制数据,仅用一个字符作为结构化标记,封装后的数据增量明显小

收稿日期:2009-10-10
作者简介:吴珊娜(1981-),女,工程硕士研究生,主要从事电子商务系统应用技术研究。

于 SOAP 协议[4]。

本文提出使用 Hessian 协议实现轻量级的 SOA 集成方案,并通过一个高校人力资源 SOA 系统整合案例说明该方案的实施过程。

1 基于 Hessian 协议的轻量级 SOA 系统集成方案

1.1 SOA 的基本结构

SOA 的基本结构主要包含 3 个角色:服务提供者、服务注册中心、服务请求者(图 1)。服务提供者提供服务,并进行注册以使服务可用,它提供处理一系列特定的服务接口,其中包含商业实体的服务或者可重用子系统的服务接口。服务注册中心起中介作用,它是服务的注册场所,汇集了大量的在线 Web 服务。服务提供者在服务注册中心发布他们的服务描述,或帮助查找 Web 服务。服务请求者通过服务注册中心查询服务地址然后再调用服务。以上 3 个角色是根据逻辑关系划的,在实际应用中,角色之间很可能有交叉,一个 Web 服务既可以是 Web 服务提供者,也可以是 Web 服务请求者,或者二者兼而有之。

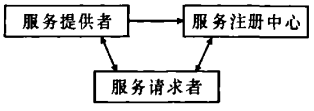

图 1 SOA 的基本结构

1.2 SOA 系统集成方案

企事业单位的信息系统中,有的信息是本单位内许多其他部门异构业务系统都需要获取的。我们提出一种基于 Hessian 协议的轻量级 SOA 系统集成方案,该方案利用 Hessian 协议,将需要对外共享的信息系统的数据和接口通过服务发布出来,供客户端通过 Hessian 协议与服务器端的服务交换数据,实现这些异构应用系统的有效整合,真正做到信息共享、及时同步。

因此我们在设计这些信息管理系统时,可以采用 SOA 的设计理念,后台服务器端采用 JAVA/J2EE 架构,业务逻辑在服务器端实现。普通用户则提供 B/S 应用界面。业务员办公前台使用 C# 实现 WinForm 富客户端开发,C# 富客户端通过 Hessian 协议访问后台的 J2EE 业务逻辑。需要提供信息整合业务的系统为其它部门对公业务提供 Web 服务接口,其它部门通过 Hessian 协议访问部署在服务器上的服务,通过服务获得需要共享的信息,实现异构系统的远程数据共享。

2 方案的实施步骤及其实现技术

我们结合一个高校人力资源 SOA 系统整合案例说明基于 Hessian 协议的 SOA 系统关键实现技术。高校图书馆的 C#.NET 富客户端要为教师办理借书证业务,教师的姓名、所在部门等真实的员工信息需要从学校人事处的人力资源管理系统中获取,这是一个远程 SOA 应用。我们通过 Hessian 协议 Web 服务实现此 SOA 应用,首先在服务端为图书馆实现一个查询服务,给此服务传入职工姓名后查询到该职工的职称信息并返回到图书馆的 C# 富客户端显示。人力资源管理系统服务器端采用 Tomcat6.0 服务器,使用 J2EE 技术实现服务,关键实施步骤如下。

2.1 在服务器端创建服务接口

首先把 Hessian 协议的 Java 版库文件 hessian.jar 添加到 Tomcat 服务器和编译路径中。此案例中对外服务接口的 Java 版定义如下:

```
public interface Profession {
 //传入字串类型的姓名,返回字串类型的职称信息
public String getProfession (String name);
}
```

此接口是一般的 Java 接口,不需要继承其它父接口。服务接口的主要作用是声明本业务逻辑中可供调用的服务方法,并定义服务方法的形参和返回参数的数据类型。

2.2 编写服务接口的实现类

上一步中的接口只是声明服务方法,没有给出服务的具体实现,所以需要编写一个接口实现类,把第一步接口中声明的抽象服务方法给予具体的实现。接口实现类的 Java 版实现如下:

```
package com.mis;
    public class ProFSeriviceImpl extends HessianServlet implements ProFService {
//服务实现类,实现接口中定义的业务逻辑函数
    public String getProfession (String name) {
//对客户端传入的姓名进行合法性校验,构造 SQL 语句
String profession = " select EmpPorF from Empbrowse where EmpName = '"+name+"' ";
// ..... 通过 JDBC 执行 SQL 语句,完成数据库查询,结果存储在 result 变量中 .....
```

```
return result;  //获取员工职称并返回
}
}
```

此实现类只是实现第一步接口中的抽象方法，它需要继承其它 hessian. jar 中的 HessianServlet 作为父类，成为一个 Hessian 业务实现类。

创建服务接口和编写服务接口的实现类相当于实现了 SOA 的服务提供者。

2.3 注册服务

Hessian 要求远程服务以 Servlet 程序的形式对外暴露出来，所以必须在 web. xml 配置文件中对 SOA 服务进行配置，配置的关键代码如下：

```
<servlet>
<! --配置 Servlet 别名，以及根据该名完成远程服务映射 -->
<servlet-name>ProFService</servlet-name>
<servlet-class>com. mis. ProFSeriviceImpl </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>ProFService</servlet-name>
<! -- 远程服务 URL 名为 ProFService  -->
<url-pattern>/ProFService</url-pattern>
<! -- 映射 Servlet 的 url -->
</servlet-mapping>
```

注册服务相当于完成了 SOA 的注册工作，重启 Tomcat 服务器后此服务程序会被自动加载并进入服务监听状态，客户端程序的调用形式为：http://ip：port/ProFService，其中的“ip”是服务器的 IP 地址，“port”是 Tomcat 的 HTTP 端口号。

2.4 客户端调用 SOA 服务

客户端远程调用服务端的服务，首先把 Hessian 协议的 C# 版库文件 hessian. dll 添加到客户端系统和 C# 编译环境中，再在客户端生成一个 C# 版的服务接口类，它与第一步中的接口作用是一样的，只是适用于 C# 语言。客户端的服务接口类充当远程 Web 服务在本地的代理(stub)，接口的代码如下：

```
using System
namespace EmpProF{
public interface ProFServcie{
   string getProfession (string name);
}
}
```

在客户端查询职称的按钮 button1 的 Click 单击事件中添加如下代码：

```
Private void button1 _ Click (object sender, System. EventArgs e){
    CHessianProxyFactory factory = new CHessianProxyFactory();//获取 Hessian 协议的工厂类
    String url = " http://127. 0. 0. 1: 8080/ProFService" ;
    //远程服务的 URL
    //通过工厂类访问远程 Hessian 服务
    ProFService P = (ProFService)factory. Create (typeof(ProFService), url);//传入员工的姓名，获取返回的职称
    String profession = P. getProfession (empname);
    MessageBox. show(profession);//在 C# 的 win 窗体中显示结果。
}
```

至此基本上完成了一个 SOA 案例的具体实施，实现了远程异构 SOA 系统集成。

从上述实施过程上看，基于 Hessian 协议的 SOA 实现过程思路简单，大大减少了代码编写的工作量和程序的复杂度，有利于提高软件开发的效率，并降低了后期维护难度。

3 结束语

SOA 为企业信息化过程中解决远程异构信息系统集成提出了解决思路，使用 Hessian 协议实现 SOA 的实施方案，和复杂的 SOAP 协议相比，具有数据传输效率高、易于实现、远程网络通信能力强等优点，属于一种轻量级的 SOA 解决方案，具有较好的工程应用价值。

参考文献：

[1] Thomas Erl. SOA 概念——技术与设计[M]. 王满红，陈荣华，译. 北京：机械工业出版社，2007：5-81.

[2] 马俊，丁晓明. 基于 SOA 的异构系统集成研究[J]. 计算机工程与设计，2008，29(42)：3638-3641.

[3] Caucho Technology，Inc. Hessian binary web service protocol [EB/OL]. [2009-5-21]. http://hessian.caucho. com/.

[4] Daniel Gredler. A Benchmark by Daniel Gredler comparing Hessian and Burlap to ORMI，Java RMI，XML-RPC，and HTTPInvoker [EB/OL]. [2008-3-11]. http://daniel. gredler. net/2008/01/07/java-remoting-protocol-benchmarks/.

(责任编辑：韦廷宗)