Web服务继承及接口机制研究

崔金红 王 旭
(对外经济贸易大学信息学院 北京 100029)

摘 要 Web服务(WS)和SOA(面向服务的架构)正被广泛使用,其最重要的应用之一是用于连接各业务系统以实现企业业务流程自动化。对现有的Web服务的定义及使用的扩展做了一些探讨,将面向对象编程中的继承和接口的机制引入Web服务,这样可以大大提高传统Web服务的可扩展性、灵活性和易用性,为Web服务及SOA架构的部署和实施提供了新的发展方向和空间。

关键词 Web服务,服务接口,继承,WSDL,远程类
中图法分类号 TP311.1 **文献标识码** A

Web Service Inheritance and Interface Web Service

CUI Jin-hong WANG Xu
(School of Information Technology & Management Engineering, University of International Business and Economics, Beijing 100029, China)

Abstract Abstract Web Service (WS) and SOA (Service-oriented Architecture) are now widely used. The most important application of SOA is connecting the various operational systems that automate an enterprise's business processes. A new extension of Web service definition and implementation-Inheritance of Web service was proposed, just like the traditional inheriting mechanism of classes and interfaces in Object-oriented programming. It makes Web service development and deployment more flexible, extendable and re-usable, and brings new thoughts and strengths to the implementation of SOA.

Keywords Web service, Web service interface, Inheritance, WSDL, Remote class

1 引言

Web服务(WS)和SOA(面向服务的架构)正被广泛使用。其最重要的应用之一是用于连接各业务系统,以实现企业业务流程自动化。采用Web服务的SOA架构,使得封装了企业业务流程的服务能够被灵活部署并且易于被其它的服务或应用访问。结合Web服务的SOA提供了一个快速业务整合的解决方案,它关注共享数据及可重复使用服务,而不是关注繁杂的企业应用专用产品的集成,因此能够更迅速、更轻松地确保企业的IT投资与企业战略保持一致。

在面向对象语言中,由于类继承以及接口等的应用,大大地增强了编程的灵活性、可读性,并且极大地提高了代码的复用度。本文将继承及接口的概念引入Web服务,提出了Web服务继承以及接口Web服务,将Web服务影射到面向对象编程的类或接口,使得Web服务的编程模式更加灵活,并可以极大地提高现有Web服务的可扩展性。

2 Web服务继承机制

通过Web服务的继承机制,可以最大限度地利用现有的服务的功能、方法。Web服务开发人员只需要把精力点放在与现有Web服务功能不同部分的实现上。而其它现有Web服务已有的功能,则可以通过Web服务继承的方式来直接获得,而提供给最终用户的则是扩展了的Web服务。该Web服务也可被再次继承和扩展,以供其它有特殊需求的用户使用。

2.1 Web服务继承

在面向对象编程中,类可以从其它类中继承数据及方法,被继承的类叫作父类,而继承的类则被称为子类。

如图1所示,本文提出了父Web服务(Super Web Service)和子Web服务(Sub Web Service)的概念。含有通用方法的Web服务(如图1中的Web Service 1)被称为父Web服务,而图1中的Web服务2和3均继承自Web服务1,它们被称为Web服务1的子Web服务。

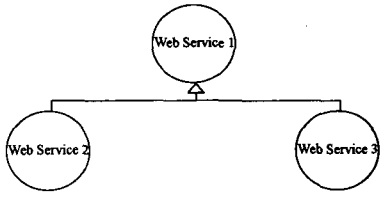

图1 Web服务继承

子Web服务可以重载父Web服务中的方法,如图2所示。当客户端调用Web服务2中的Method A,由于Web服

到稿日期:2008-07-15 返修日期:2009-02-17 本文受对外经济贸易大学"211工程"三期重点学科建设项目(33003)资助。
崔金红(1973—),女,博士,副教授,研究方向为信息系统、工作流等,E-mail: cuijinhongwx@sohu.com;王 旭(1971—),男,博士,工程师,研究方向为计算机图形学、科学可视化、计算机通信及应用、IMS/NGN及3G应用等。

务 2 继承自 Web 服务 1 而且未重载该方法，则 Web 服务 2 可将客户端的请求转发到 Web 服务 1，并可将 Web 服务 1 的请求结果转发给客户端，我们称之为 Web 服务继承的代理模式。

如图 3 所示，Web 服务 2 也可直接将客户端的请求重定向到其父 Web 服务 1，客户端直接与 Web 服务 1 交互并获得所需的服务。

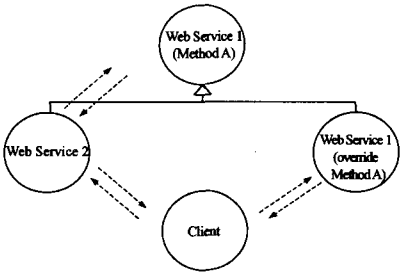

图 2　Web 服务继承的代理模式

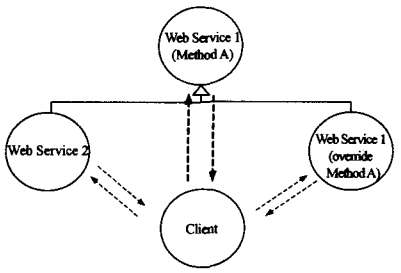

图 3　Web 服务继承的重定向模式

而对于重载的方法，则直接由子 Web 服务响应客户端的请求。

子 Web 服务也可以添加新的方法来扩展现有的父 Web 服务的功能。如图 4 所示，与面向对象的 C++编程一样，子 Web 服务可以从多个父 Web 服务继承，也可如 Java 一样限定子 Web 服务只能有一个父 Web 服务。

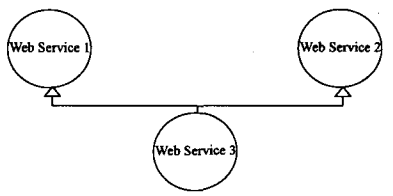

图 4　Web 服务多继承

2.2　Web Service 接口

如同面向对象编程，本文引入了 Web 服务接口的概念，目的是规范 Web 服务功能的实现，即保证不同的 Web 服务实现均提供规范的接口或方法，以供客户端调用。

如图 5 所示，所谓的 Web 服务接口，就如同面向对象编程中的抽象类一样，只有方法的定义，没有方法的实现，而方法的实现则由其对应的实体 Web 服务来完成。

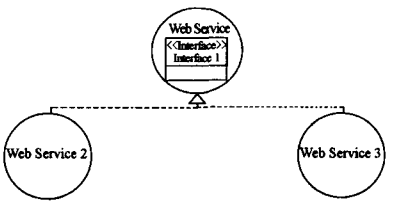

图 5　Web 服务接口

如图 6 所示，Web 服务也可以实现多个 Web 服务接口。

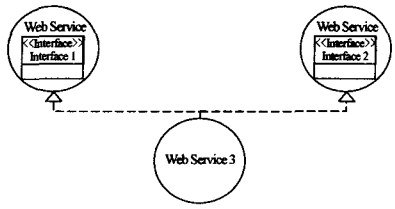

图 6　实现多个接口的 Web 服务

3　应用实例

Internet 上有一免费的天气预报 Web 服务，其中定义了天气查询的方法 GetWeather，该 Web 服务的描述节选如下：

```
〈wsdl:message name="GetWeatherSoapIn"〉
 〈wsdl:part name="parameters" element="tns:GetWeather" /〉
〈/wsdl:message〉
〈wsdl:message name="GetWeatherSoapOut"〉
 〈wsdl:part name="parameters" element="tns:GetWeatherResponse" /〉
〈/wsdl:message〉
〈wsdl:portType name="WeatherSoap"〉
 〈wsdl:operation name="GetWeather"〉
  〈wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"〉
  〈wsdl:input message="tns:GetWeatherSoapIn" /〉
  〈wsdl:output message="tns:GetWeatherSoapOut" /〉
 〈/wsdl:operation〉
〈/wsdl:portType〉
```

现在需要扩展该天气预报服务，在该天气预报服务的基础上提供摄氏和华式温度的转换功能，而扩展所采用的 Web 服务接口定义如下：

```
〈wsdl:message name="c2FRsp"〉
 〈wsdl:part element="impl:c2FRsp" name="parameters"/〉
〈/wsdl:message〉
〈wsdl:message name="f2CRsp"〉
 〈wsdl:part element="impl:f2CRsp" name="parameters"/〉
〈/wsdl:message〉
〈wsdl:message name="f2CRequest"〉
 〈wsdl:part element="impl:f2C" name="parameters"/〉
〈/wsdl:message〉
〈wsdl:message name="c2FRequest"〉
 〈wsdl:part element="impl:c2F" name="parameters"/〉
〈/wsdl:message〉
〈wsdl:portType name="ConvertTemperature"〉
 〈wsdl:operation name="c2F"〉
  〈wsdl:input message="impl:c2FRequest" name="c2FRequest"/〉
  〈wsdl:output message="impl:c2FRsp" name="c2FRsp"/〉
 〈/wsdl:operation〉
 〈wsdl:operation name="f2C"〉
  〈wsdl:input message="impl:f2CRequest" name="f2CRequest"/〉
  〈wsdl:output message="impl:f2CRsp" name="f2CRsp"/〉
 〈/wsdl:operation〉
〈/wsdl:portType〉
```

即在原天气预报 Web 服务的基础上需要添加 c2F 和 f2C

方法。如图7所示，我们可以定义一个新的combineWeather服务，其中包括从原有的天气预报Web服务中继承的GetWeather方法，以及从Web服务接口定义中引入的c2F和f2C方法。

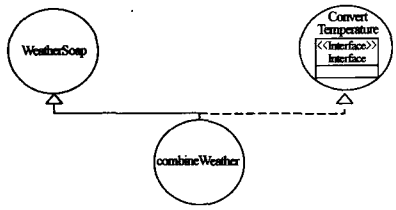

图7 Web服务应用实例

在新的combineWeather服务的实现中，GetWeather方法可由其父Web服务获得天气信息。若采用代理模式，在combineWeather服务中还可以修改WeatherSoap服务中返回的内容后再转发给其客户端。而c2F和f2C方法则需在combineWeather中实现，以供其客户端调用。新生成的combineWeather服务的方法描述大致如下：

```
〈wsdl:portType name="combineWeather"〉
 〈wsdl:operation name="GetWeather"〉
  〈wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"〉
  〈wsdl:input message="tns:GetWeatherSoapIn" /〉
  〈wsdl:output message="tns:GetWeatherSoapOut" /〉
〈/wsdl:operation〉
〈wsdl:operation name="c2F"〉
  〈wsdl:input message="impl:c2FRequest" name="c2FRequest"/〉
  〈wsdl:output message="impl:c2FRsp" name="c2FRsp"/〉
 〈/wsdl:operation〉
 〈wsdl:operation name="f2C"〉
  〈wsdl:input message="impl:f2CRequest" name="f2CRequest"/〉
  〈wsdl:output message="impl:f2CRsp" name="f2CRsp"/〉
 〈/wsdl:operation〉
〈/wsdl:portType〉
```

4 IDE扩展

可以增强Web服务集成开发环境，以支持Web服务的继承和接口Web服务。当用户需要继承某个或某几个Web服务或实现Web服务接口时，需要指定所继承的Web服务或所要实现接口的描述文件或描述文件的URL，并指定所采用的Web服务继承模式是代理模式还是重定向模式；然后，用户可以选择重载父Web服务的一些方法，IDE可以自动产生这些方法的框架代码。对于那些未被重载的方法，IDE可以根据用户指定的Web服务的继承模式自动生成服务代码；如图8所示，IDE还可根据Web服务所要实现的接口生成接口方法的框架代码。如同目前通用的Web服务集成开发环境一样，IDE还可根据当前Web服务所继承的Web服务描述、实现的Web服务接口以及用户新添加的方法定义生成新的Web服务描述文件。

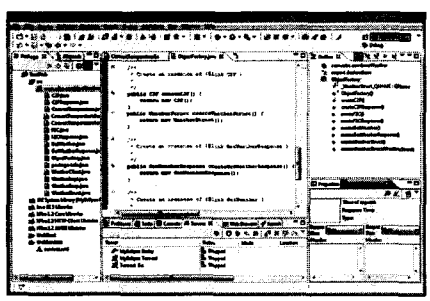

图8 Web服务集成开发环境

结束语 本文对传统的Web服务做了开创性的扩展，通过继承以及Web服务接口概念的引入，使得Web服务更像是面向对象编程中的类或对象，只不过该类或对象是远程的，而Web服务的使用则类似于远程对象方法的调用。通过将面向对象编程的特征引入Web服务以及SOA，可以大大提高传统Web服务的可扩展性、灵活性和易用性，为Web服务及SOA架构的部署和使用提供了新的发展方向和空间。

参考文献

[1] Gudgin M, Hadley M, Mendelsohn N. SOAP Version 1.2 Parts 1-2[R]. W3C Recommendation, World Wide Web Consortium, June 2003

[2] Christensen E, Curbera F, Meredith G. Web Services Description Language (WSDL) 1.1[R]. W3C Note, World Wide Web Consortium, March 2001. (A W3C Recommendation for WSDL 2.0 is currently pending)

[3] Duce D. Portable Network Graphics (PNG) Specification (Second Edition)[R]. W3C Recommendation, World Wide Web Consortium, November 2004

[4] McGuinness D L, van Harmelen F. OWL Web Ontology Language Overview[R]. W3C Recommendation, World Wide Web Consortium, February 2004

[5] Szyperski C. Component software: Beyond Object-oriented programming[M]. Addison-Wesley, 1998

[6] Nabor MC, Silva J A F, Anido R O. Client-side selection of replicated web services: An empirical assessment[J]. Journal of Systems and Software, 2008, 81(8):1346-1363

[7] Andres Q, Manish P. A framework for distributed content-based web services notification in Grid systems[J]. Future Generation Computer Systems, 2008, 24(5): 452-459

[8] Serena P. The service discovery methods issue: A web services UDDI specification framework integrated in a grid environment [J]. Journal of Network and Computer Applications, 2008, 31(2):93-107

[9] Zakaria M, Djamal B, Philippe T, et al. Towards a context-based multi-type policy approach for Web services composition [J]. Data & Knowledge Engineering, 2007, 62(2):327-351

[10] Sangyoon O, Fox G C. Optimizing Web Service messaging performance in mobile computing[J]. Future Generation Computer Systems, 2007, 23(4):623-632

[11] Claudio G, Roberto L, Manuel M. A Formal Framework for Web Services Coordination[J]. Electronic Notes in Theoretical Computer Science, 2007,26,180(2):55-70

[12] van Aalst der W MP, Boualem B, Fabio C, et al. Business process management: Where business processes and web services meet[J]. Data & Knowledge Engineering,2007,61(1):1-5

(下转第191页)

$$召回率=\frac{正确分类的数量}{预先标记的数据比率}$$

综合两个指标，可以得到一个新的综合评价指标，即 $MicroF1=\frac{2(precision*recall)}{precision+recall}$。我们采用 AMB[10] 和 CHI 两种特征选择算法和本文中给出的算法进行比较、分析。

如图 2 和图 3 所示，当取 10 个特征子集数据进行分类时，DPFS 算法的 *MicroF*1 值小于 CHI 方法的 *MicroF*1 值，但是随着特征的增加，DPFS 结果子集表现越好，当特征数为 1000 时，3 种方法的 *MicroF*1 值基本相同，但是用 DPFS 算法的结果子集进行分类时，会取得很好的结果。在实验中，我们发现当 CHI 方法在特征数达到 10000 时，分类效果出现了降低，这说明了特征为数不断加大，促使大量无关特征和冗余特征的产生，导致分类性能的下降。

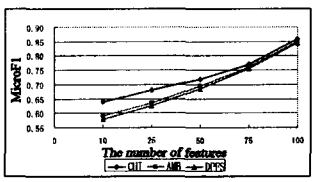

图 2　对低维数据分类的 MicroF1 值比较

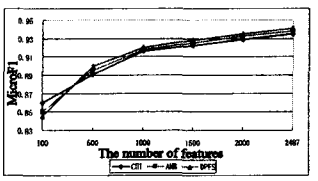

图 3　对高维数据分类的 MicroF1 值比较

在对分类效果做过测试之后，我们又对 3 种算法的运行时间作了比较测试，如图 4 所示，子集特征较少的情况下 CHI 所需要的时间明显要少于其他两种算法，但随着子集特征的维数的增加，本文提出的 DPFS 算法体现出了其优越性，因为本文提出的 DPFS 算法结合了动态规划的思想，即保存已解决的子问题的答案，而在需要时再找出已求得的答案，这样就可以避免大量重复计算，提高运行时间。

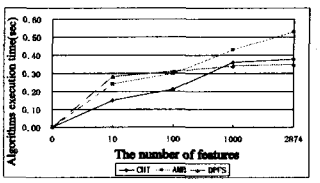

图 4　算法执行时间的比较

综合上述两个实验结果，我们可以看出，传统的 CHI 特征选择算法在选择特征较小时，能够选择出一部分较好的特征，但是不能选择出近似最优的特征子集。AMB 算法虽然能够在这个集合中选择出近似最优的特征子集，但计算量较大。因此可以看出，文本提出的 DPFS 算法不仅能选择出近似最优特征，还通过减少计算量来提高算法的运行时间。

结束语　本文提出的一种基于动态规划思想的特征选择算法，与传统的特征选择算法不同，它能更为全局地考虑，去掉冗余的特征，得到一个近似最优的特征子集。由于其加入了动态规划的思想，在算法的运行时间上也有了一定程度的提高，从实验来看，本文提出的算法与其他算法相比较，有一定的优势。对于将来的工作，我们可能将对本算法的相关值的存储结构进行优化以及将其应用在 Web 挖掘中做一些工作。

参 考 文 献

[1] Greengrass E. Information retrieval: A surrvey[R]. DOD. Maryland. 2000

[2] Yang Yiming. Pedersen J O. A Comparative Study on Feature Selection in Text Categorization [C] // Anon. Proceedings of 14th International Conference on Machine Learning(ICML-97). Nashville: TN, 1997: 412-420

[3] He Jing-song, Shi Ze-sheng. Method of feature selection using signal analysis[J]. Journal of University of science and Technology of China, 2001

[4] Yu L, Liu H. Efficient Feature Selection via Analysis of Relevance and Redundancy[J]. Journal of Machine Learning Research, 2004, 10: 1205-1224

[5] Qu G Z, Hariri S, Yousif M. A New Dependency and Correlation Analysis for Feature[J]. IEEE Trans. on Knowledge and Data Engineering, 2005, 17: 1199-1207

[6] Yu L, LIU H. Feature selection for high-dimensional data: a fast correlation-based filter solution[A] // Proceedings of the Twentieth International Conference on Machine Learning[C]. Washington, 2003: 856-863

[7] Yan J, Liu N, Zhang B, et al. OCFS: Optimal orthogonal centroid feature selection for text categorization[C] // Proceedings of the ACMSIG on Information Retrieval. Salvador, Brazil, 2005: 122-129

[8] 王晓东，计算机算法设计与分析(第三版)[M]. 北京：电子工业出版社，2007

[9] Porter. An Algorithm for Suffix Stripping Program[J]. 1980, 14(3): 130-137

[10] Cui Zi-feng, Xu Bao-wen, Zhang Wei-feng, et al. An Approximate Markov Blanket Feature Selection Algorithm[J]. 2007, 30(12): 2074-2081

(上接第 146 页)

[13] Foukarakis I E, Kostaridis A I, Biniaris C G, et al. Webmages: An agent platform based on web services[J]. Computer Communications, 2007, 30(3): 538-545

[14] Therani M, Uttamsingh N. A declarative approach to composing web services in dynamic environments[J]. Decision Support Systems, 2006, 41(2): 325-357

[15] Zakaria M. On coordinating personalized composite web services [J]. Information and Software Technology, 2006, 48(7): 540-548

[16] Hwang San-Yih, Lim Ee-Peng, Lee Chien-Hsiang, et al. On Composing a Reliable Composite Web Service: A Study of Dynamic Web Service Selection[C] // IEEE International Conference on Web Services. 2007: 184-191