View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems

AIS Electronic Library (AISeL)

ICEB 2010 Proceedings

International Conference on Electronic Business (ICEB)

Winter 12-1-2010

Research and Implementation of Web Service Inheritance and Interface Web Service

Jinhong Cui

Wang Xu

Follow this and additional works at: https://aisel.aisnet.org/iceb2010

This material is brought to you by the International Conference on Electronic Business (ICEB) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICEB 2010 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

RESEARCH AND IMPLEMENTATION OF WEB SERVICE INHERITANCE AND INTERFACE WEB SERVICE

Jinhong Cui, Department of Information Management, School of Information Technology & Management Engineering, University of International Business and Economics, Beijing, China.
E-mail: cjh1616@126.com
Xu Wang, IBM CDL, Beijing, China.
E-mail: xuwang@ibm.com.cn

Abstract

Web Service (WS) and SOA (Service-Oriented Ar-chitecture) are now widely used. The most important application of SOA is connecting various business systems that automate an enterprise's business processes. A new extension of Web Service definition and implementation-Inheritance of web service is pro-posed in this paper, 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.

1. Introduction

Web Service (WS) and SOA (Service-Oriented Ar-chitecture) are now widely used. The most important application of SOA is connecting various business sys-tems that automate an enterprise's business processes. SOA with Web services enables the development of services that encapsulate business functions and that are easily accessible from any other service. The com-bination of Web services and SOA provides a rapid integration solution that more quickly and easily aligns investments and corporate strategies by focusing on shared data and reusable services rather than proprie-tary integration of products.
In object-oriented programming languages such as C++ and Java, by introducing Class inheritance and interface, greatly enhanced the flexibility, readability of source code, and significantly improve the code reusability.
This paper introduces the concept of Web services inheritance, as well as Interface Web services. Web services will be mapped to class or interface of the ob-ject-oriented programming language. This approach makes Web services programming more flexible and greatly improve scalability of current web service.

2. Mechanism of Web Service Inheritance

In this paper, Inheritance of web service is proposed just like the traditional inheriting mechanism of classes and interfaces in Object-oriented programming.
It provides a novel way for web service design and development. It allows web service inheritance so to maximize reuse existing features/methods of web ser-vices. Users only need to focus on the fea-tures/methods/interfaces differentiation of the new web services. It is more flexible, extensible and reusable. It is web service level integration, through the use of corresponding code generation tool of IDE(e.g., Eclipse plugin), users don't have to know the details of the super web services.

2.1 Inheritance of web service

Object-oriented programming allows classes to in-herit commonly used data and methods from other classes. These inherited and inheriting classes are called the superclass and the subclasses respectively.
As described in Figure 1, in this paper, web service (web service 1) with commonly used methods and data are called the superwebservice, and those (web service 2 and 3) inherited from the superwebservice are called subwebservice.

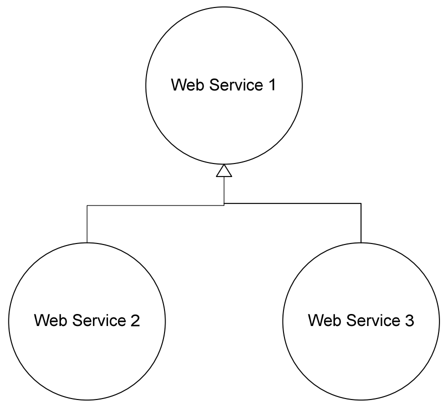


Figure 1. Inheritance of web service

The subwebservice can override the methods defini-tion in the superwebservices. As described in the bel-lowing diagram, when web client calls the method A that is not overridden by web service 2, this request can be forwarded to the superwebservice web service 1, and the method A in web services 1 will be called and the result will also be forwarded by web service 2 to the web client (referring to Figure 2). Web service 2 can also redirect the client's web request directly to web service 1 as shown in Figure 3.

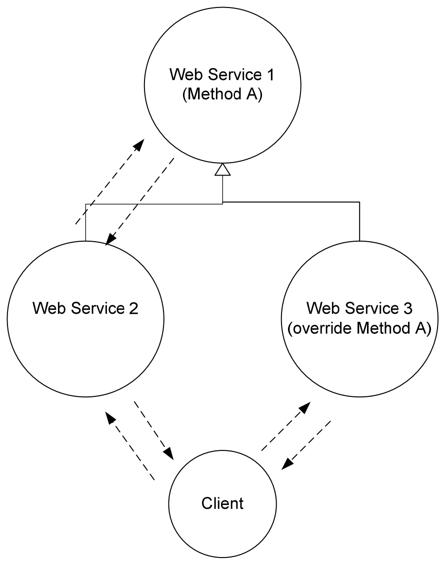


Figure 2. Proxy model of web service inheritance

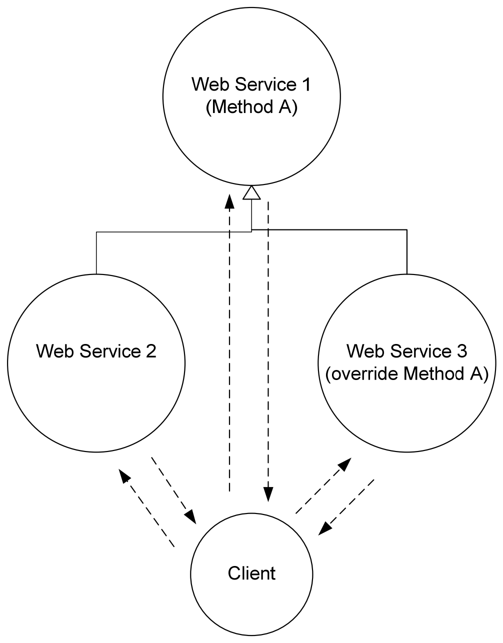


Figure 3. Redirection model of web service inheritance

Subwebservice can also add new methods to extend the functions of the existing superwebservice.
Just like Object-oriented C++ programming, sub-webservice can inherit from multiple superwebservice as described in Figure 4. Further more, subwebservice can also be limited to only be able to inherit from only one superwebservice like Java does.

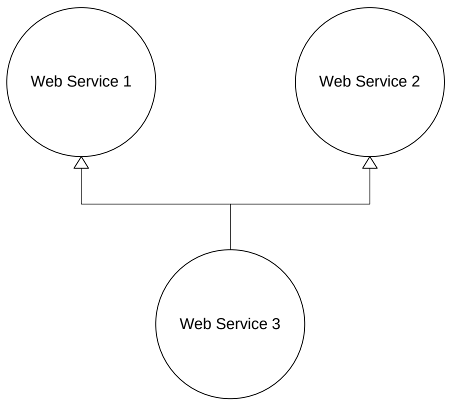


Figure 4. Multiple inherit of web service

2.2 Interface web service

Just like Object-oriented programming, a so-called interface web service can be introduced to guarantee the desired web methods be generated in its implemen-tation or subwebservice as described in Figure 5.
Actually, the so-called interface web service is just empty methods definition set without implementation, just like the abstract methods definition in class.

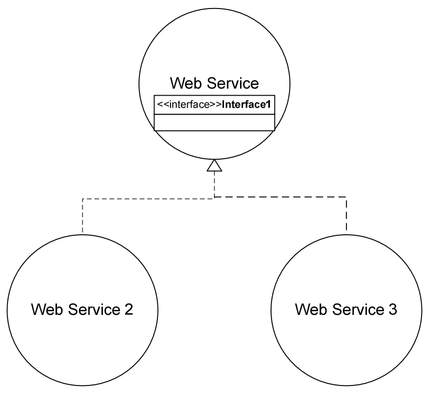


Figure 5. Interface web service

As described in the Figure 6, web service can also implement multiple interface web services.

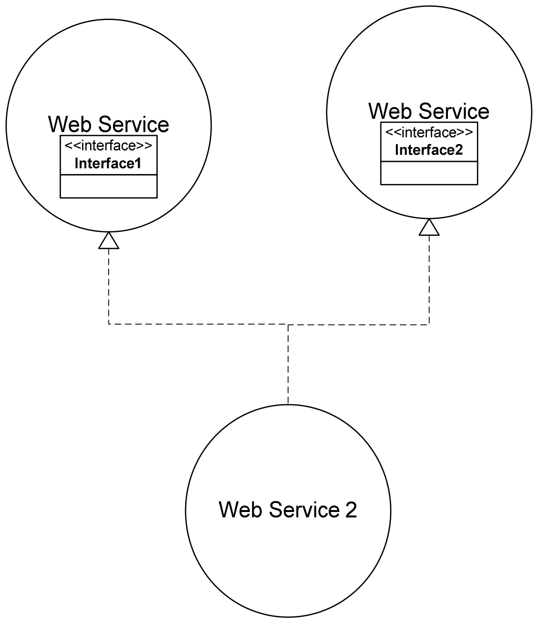


Figure 6. Web service with multiple interface

3. Implementation

Introduce the paper with an abstract of approximately 100 words. Begin in the left column with centered heading "Abstract" set above the single-spaced abstract text. The abstract should properly describe the findings or arguments presented in the paper.
There is A free weather forecast Web service from internet, which defines a method of inquiry weather GetWeather, the main part of the Web service description of it as bellowing:

```
<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>
```

This above weather forecast web service is needed to be extended to support Fahrenheit and Celsius temperature conversion function, and the description of the interface web service convertTemperature used to extend the weather forecast Web service as bellowing:

```
<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>
```

In the above web service description, c2F and f2C methods will be added to the original weather forecast Web service. As shown in Figure 7, we can define a new combineWeather services, including weather forecasts function inherited from the original Web service GetWeather, as well as c2F and f2C methods from the definition of Web services interface convertTemperature.

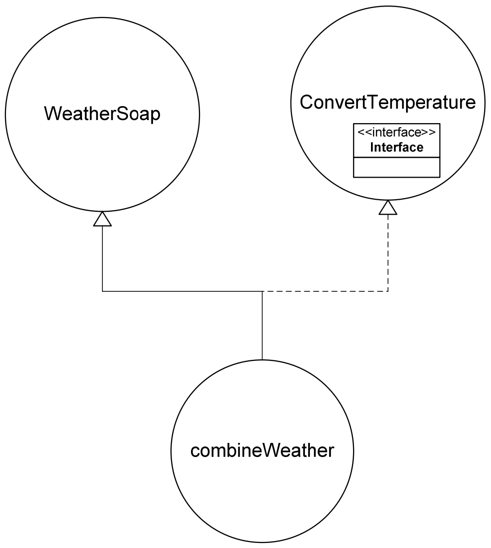


Figure 7. Sample web service

In the implementation of this new combineWeather services, GetWeather method can be inherited from its parent web service to obtain weather information. If use proxy model, combineWeather services can also modify the result gotten from the WeatherSoap web service and return the modified weather information to the client, and c2F and f2C methods will be implemented in CombineWeather web service. The description of the new CombineWeather Web service as bellowing:

```
<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>
```

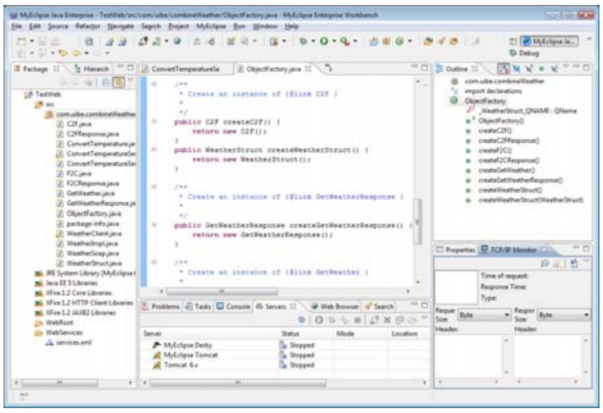
4. IDE extension

Traditional web service IDE (Integrated develop-ment environment) can be enhanced to support web service inheritance and interface web service.

When user needs to inherit one or more web service or implement one or more interface web service , he/she just need to give the URLs of the description of them, and specify if proxy Mode or redirect mode will be used; then, the user can also choose to over-ride some methods from its parent Web service. Then the enhanced IDE will automatically generate the frame-work of the code of these methods, for those un-over-ride methods, IDE will automatically generate the web service code according to user-chosen inherited model.

As shown in Figure 8, IDE can also generate the code framework for methods defined in according interface definitions.

The same as current popular Web services inte-grated development environment, IDE can also auto-matically generate web service description file for the new web service based on the description file of its parent web service and implemented interface web services.



```
    /**
     * Create an instance of {@link C2F }
     *
     */
    public C2F createC2F() {
        return new C2F();
    }
    public WeatherStruct createWeatherStruct() {
        return new WeatherStruct();
    }

    /**
     * Create an instance of {@link GetWeatherResponse }
     *
     */
    public GetWeatherResponse createGetWeatherResponse() {
        return new GetWeatherResponse();
    }

    /**
     * Create an instance of {@link GetWeather }
     *
```

```
com.uibe.combineWeather
import declarations
ObjectFactory
    _WeatherStruct_QNAME : QName
    ObjectFactory()
    createC2F()
    createC2FResponse()
    createF2C()
    createF2CResponse()
    createGetWeather()
    createGetWeatherResponse()
    createWeatherStruct()
    createWeatherStruct(WeatherStruct)
```

Server	Status	Mode	Location
MyEclipse Derby	Stopped		
MyEclipse Tomcat	Stopped		
Tomcat 6.x	Stopped		

Figure 8. Web service IDE

5. Summary

Through the introduction of Web service inheritance and Interface Web service, this paper extend the web service definition; make web service definition more flexible, extendable and reusable, and bring new thoughts and strengths to the development and imple-mentation of SOA.

It makes web service more like remote classes or in-terface as the traditional Object-oriented program-ming, and bring more features and advantages of Ob-ject-oriented mechanism into web service and SOA design and development. It allows web service inheri-tance so to maximize reuse existing features/methods of web services. Developers only need to focus on the differentiated features/methods/interfaces of the new web services and don't have to learn the details of the super web services.

References

[1] M. Gudgin, M. Hadley, N. Mendelsohn et al. SOAP Version 1.2 Parts 1–2. W3C Recommendation, *World Wide Web Consortium*, June 2003.

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

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

[4] D. L. McGuinness & F. van Harmelen. OWL Web Ontology Lan-guage Overview. W3C Recommendation, *World Wide Web Consor-tium*, February 2004.

[5] C. Szyperski. Component software: Beyond Object-oriented programming.

Addison-Wesley, 1998.

[6] Nabor C. Mendonça, José Airton F. Silva, Ricardo O. Anido. Client-side selection of replicated web services: An empirical assessment. *Journal of Systems and Software*, August 2008, 81(8), pp.1346-1363

[7] Andres Quiroz, Manish Parashar. A framework for distributed content-based web services notification in Grid systems. *Future Generation Computer Systems*, May 2008, 24(5), pp.452-459

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

[9] Zakaria Maamar, Djamal Benslimane, Philippe Thiran, Chirine Ghedira, Schahram Dustdar, Subramanian Sattanathan. Towards a con-text-based multi-type policy approach for Web services composition. *Data & Knowledge Engineering*, August 2007, 62(2), pp.327-351

[10] Sangyoon Oh, Geoffrey C. Fox. Optimizing Web Service messaging performance in mobile computing. *Future Generation Computer Systems*, May 2007, 23(4), pp.623-632

[11] Claudio Guidi, Roberto Lucchi, Manuel Mazzara. A Formal Framework for Web Services Coordination. *Electronic Notes in Theoretical Computer Science*, 26 June 2007, 180(2),pp. 55-70

[12] Wil M.P. van der Aalst, Boualem Benatallah, Fabio Casati, Francisco Curbera, Eric Verbeek. Business process management: Where business processes and web services meet. *Data & Knowledge Engineering*, April 2007, 61(1), pp.1-5

[13] I.E. Foukarakis, A.I. Kostaridis, C.G. Biniaris, D.I. Kaklamani, I.S. Venieris. Webmages: An agent platform based on web services. *Computer Communications*, 2 February 2007, 30(3), pp.538-545

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

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

[16] San-Yih Hwang; Ee-Peng Lim; Chien-Hsiang Lee; Cheng-Hung Chen. On Composing a Reliable Composite Web Service: A Study of Dynamic Web Service Selection. IEEE International Conference on Web Services, 2007, 9-13 July 2007, pp.184 - 191