Web Services Example with PHP/SOAP

Martin Tsenov

Abstract: *This paper explains decision for Web Services architecture between distributed network systems. The proposed method is based on Open Source standards - SOAP and WSDL. Software solution based on the proposed architecture, developed using SOAP extension for PHP is presented and explained.*

Key words: *PHP, SOAP, Web Services*

INTRODUCTION

The use of data exchange and need to define resource on the World Wide Web is expanding rapidly with the application-to-application communication and interoperability grows. These services provide a standard means of network communication between different software applications involved in presenting dynamic context-driven information to the user. In order to promote interoperability and extensibility among these applications, as well as to allow them to be combined in order to perform more complex operations, Web services architecture is needed. In this paper the author describes a set of requirements for Web Services architecture between distributed network systems and developed software implementation example, based on the proposed method. Web Services example is based on the SOAP protocol, WSDL standard and SOAP extension for PHP. The main aim of the proposed method is to provide data access and exchange to various clients via Web Services. The example revolves around the open source standards, which allows for clients to access and exchange data and resources between distributed databases. It also allows various other system users to interact with the application. Below is the sequence of activities of the example:

- Define the Web services architecture.
- Implementation of the Web services architecture.
- Develop a Software implementation based on the Web services architecture.

WEB SERVICES ARCHITECTURE AND SOFTWARE IMPLEMENTATION

A. Theoretical part

- SOAP - SOAP (Simple Object Access Protocol) [1] is a lightweight XML-based protocol for exchanging structured information between distributed applications over native web protocols, such as HTTP. SOAP specifies the formats that XML messages should use, the way in which they should be processed, a set of encoding rules for standard and application-defined data types, and a convention for representing remote procedure calls and responses.

SOAP protocol consists of three parts:

1. An envelope which describes the contents of the message and how to use it
2. A set of rules for serializing data exchanged between applications
3. A procedure to represent remote procedure calls, that is, the way in which queries and the resulting responses to the procedure are represented.

- WSDL - WSDL (Web Service Description Language) [2] is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.

B. Web services architecture.

The Web services architecture is defined by the following steps:

- The calling application Internet browser (IE, Mozilla) or other Application (Web program) makes a procedure call on the WSDL file and SOAP Service client.
- The SOAP Service client takes the method and parameters and builds an XML container for them; the XML container is the sent over HTTP as a SOAP request.
- SOAP Service server receives the SOAP requests; Soap_parser_class parses that XML container and determines the method to be called and the parameters to this method.
- The method is executed on the server and returns a result.
- The result is packaged as XML and the server returns the XML result container as the response of the POST request by Soap_transport_http_class.
- The client parses the XML response container and returns the result to the calling application.
- The application processes the result.

Figure 1 presents the structure of the Web services architecture.

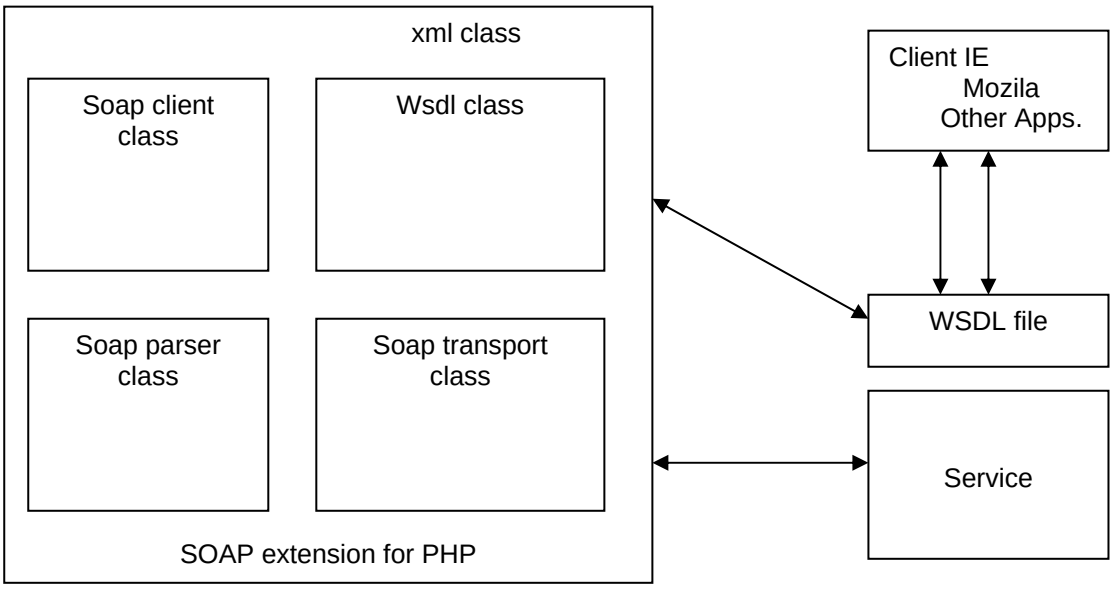

Fig.1 Web services architecture

C. Software implementation of Web services architecture.

Software implementation of the Web services [3] architecture is used for conference web system for submission, upload and review. As a programming language open source CGI script language PHP5 [4] is used. The example is defined by the following steps:

1. To define SOAP Client in the beginning of the Soap Service, it is need to gather some information about this particular service [5]:

- The method name
- The endpoint URL where the service is running
- The SOAPAction header value for the method
- The namespace URI for the method
- Input and output parameter names and types

In the proposed exampled the needed information are display below:

Table1. Service Definition

Method Name	getData
Endpoint URL	http://hs19.iccs.bas.bg/nusoap
SOAPAction	urn:getData#getData
Method Namespace URI	urn:getData
Input Parameters	Symbol:string
Output Parameters	Result:float

The service information is implemented in the example by SOAP Service client written in PHP5 [5]: Table 2 display SOAP Service client class

Table2. SOAP Service client

```
<?php
$client = new SoapClient(NULL,
        array( "location" => "http://hs19.iccs.bas.bg/nusoap",
                  "uri" => "urn:getData",
                "style" => SOAP_RPC,
                 "use"  => SOAP_ENCODED
              ));
……..
……..
print($client->__call(
        /* SOAP Method Name */
        /* Parameters */
        array(
            new SoapParam(
                /* Parameter Value */
                /* Parameter Name */
        )),
        /* Options */
        array(
            /* SOAP Method Namespace */
            /* SOAPAction HTTP Header for SOAP Method */
        )). "\n");
……..
……..
?>
```

2. The second task is to create a WSDL document [6] describing our service in a format that client requests will understand. Below is the structure of WSDL document with two messages (sub.wsdl):

The *message* section defines two messages. The first is *getDataReques*t, which is a request to relay the *getData* message and takes one string parameter called *symbol*. The other is *getDataResponse*, which is a response to the *getData* message, containing one float value, named *Result*. The *portType* section defines one operation, *getData*, which describes which of the *messages* listed in the message section will be used to transmit the request and response. The *binding* section defines how the messages must be transmitted and encoded. Here it tells us that we will be sending an RPC request using SOAP encoding across HTTP. It also specifies namespace and value of the SOAPAction header for the *getData* method. The *service* section defines the endpoint URL where the service is running.

Table 3 display the structure of WSDL document.

Table3. Structure of WSDL document

```
<?xml version ='1.0' encoding ='UTF-8' ?>

<definitions name='getData'
   targetNamespace='getData'
   xmlns:tns='getData'
   xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
   xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
   xmlns='http://schemas.xmlsoap.org/wsdl/'>

<message name='getDataRequest'>
  <part name='symbol' type='xsd:string'/>
    </message>
<message name='getDataResponse'>
  <part name='Result' type='xsd:float'/>
    </message>

<portType name='getDataPortType'>
  <operation name='getData'>
    <input message='tns:getDataRequest'/>
    <output message='tns:getDataResponse'/>
  </operation>
</portType>

<binding name='getDataBinding' type='tns:getDataPortType'>
   <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http'/>
   <operation name='getData'>
   <soap:operation soapAction='urn:getData#getData'/>
     <input>
       <soap:body use='encoded' namespace='urn:getData'
       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
     </input>
     <output>
       <soap:body use='encoded' namespace='urn:getData'
       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
     </output>
   </operation>
</binding>

<service name='getDataService'>
  <port name='getDataPort' binding='getDataBinding'>
    <soap:address location='http://hs19.iccs.bas.bg/nusoap'/>
  </port>
</service>

</definitions>
```

3. The third task of example is the development of Soap_parser_class, Soap_transport_http_class and integration [7] of the Web system for submission, upload and review with proposed Web Services architecture. All of the files of the Web system must be located in the *service* section, defined in wsdl file, where the endpoint URL of the service is running.
The software example can be reach at http://hs19.iccs.bas.bg/nusoap/ws_wizard.php.

C. Comparison between Open Sources based software tools and commercial (shareware) software products SAP NetWeaver and IBM's Web Services Process Management Toolkit.

1. Review of commercial (shareware) software products.

1.1 SAP NetWeaver [8]:

- Technical Details: SAP NetWeaver is designed as a platform for the development, deployment, and management of Web services. For example, SAP NetWeaver Portal uses Web services to offer role-based interfaces that allow partners to work together. The application platform provides existing application functionality, based on Java or ABAP, as Web services. These services are described using Web Services Description Language (WSDL).

The Web services are provided through a framework used by the integration broker to describe Web services interfaces in a Universal Description, Discovery, and Integration (UDDI) repository. Business process management coordinates the activities of Web services provided by business partners to manage processes across applications.

Simple Object Access Protocol (SOAP) provides a mechanism for sending Web service messages and XML service requests based on WSDL.

1.2 Web Services Process Management Toolkit – part of the IBM's WebSphere MQ Workflow [9]:

- Technical Details: The Web Service Process Management Toolkit (WSPMTK) combines business process management technology with Web Services and offers the tools and samples needed. Web Service Process Management Toolkit working process description:
  - *Compose Web Services into a business process*: Composing Web Services allows developers to choreograph the interaction of a set of Web Services within a business process and add control logic to the business process.
  - *Implement a Web Service as its own business process*: Using a process as implementation for a Web Service allows developers to compose complex Web Services with the characteristics of a process.
- Software requirements:
  - JDK V1.3.1or later.
  - WebSphere MQ Workflow V3.4 or later.
  - Apache AXIS V1.1
  - Web Application Server, which complies with the Java Servlet V2.2 or later.

2. Conclusions based on the comparison between Open Sources based software tools and commercial (shareware) software products - the advantages of using Open Source based products and standards:

- Possibility for free integration with others products based on the Open Source.
- Possibility for free integration of scientific results (algorithms, methods, models).
- Possibility for future improvement of the work.
- No need of software licenses (freeware).

RESULTS

Main results of the paper:

- Presented Web services architecture: describes algorithm for Web services architecture based on the WSDL, SOAP and XML standards.
- Develop and describe WSDL file for Web Services.
- Develop software application based on the Open Source Web services architecture.

CONCLUSIONS AND FUTURE WORK

This paper explained decision for Web Services architecture between distributed network systems. The proposed method is based on Open Source standards - SOAP and WSDL. Software solution based on the proposed architecture, developed using SOAP extension for PHP is presented and explained. The main aim of the proposed model is to provide data access to various clients via Web Services. The example revolves around the open source standards, which allows for clients to access and exchange data and resources between distributed databases. It also allows various other system users to interact with the application.

The future work is about the problem for optimization of searching methods for services and resources.

REFERENCES

[1] http://www.w3.org/TR/2003/REC-soap12-testcollection-20030624/

[2] http://www.developer.com/services/article.php/1602051

[3] Stoilov T., K. Stoilova. Integration of Web Services in Internet. 18th International Conference on Systems for Automation of Engineering and Research "SAER-2004", 24-26 September, 2004, St. Konstantin resort, Varna, Bulgaria.

[4] C. Scollo, J. Castagnetto, D. Veliath, H. Rawat, S. Schumann, Professional PHP Programming, Wrox, December 1999

[5] http://www.zend.com/php5/articles/php5-SOAP.php

[6] Newcomer E., Understanding Web Services: XML, WSDL, SOAP and UDDI, Pearson Education, 2002

[7] Ivanova E., Application of Distributed Search in Databases for Web Services, 2003, Proceedings of International conference ICEST'03, Sofia, Bulgaria, p.291-294

[8] http://www.sap.com/platform/netweaver/technicaldetails/webservices.epx

[9] http://www-306.ibm.com/software/websphere/

ABOUT THE AUTHOR

Assist. Prof. Martin Tsenov, Department Hierarchical Systems, Institute of Computer and Communication Systems - Bulgarian Academy of Sciences, Sofia, Phone +359 2 979 2774, E-mail: mcenov@hsi.iccs.bas.bg

This research is partly supported by the European Commission, project №FP6-027178.