Web Services as new phenomenon in the PHP environment

Pavel Horovčák[1]

Abstract

The support of development and exploitation of Web Services (WS) is gradually becoming an integral part of current development environments. Beside standard environments connected with the emergence of WS (Java or .NET), the support is presently time realized also in a widely-used environment for the web application development – PHP, in its updated version 5. This contribution is oriented towards the development and utilization of WS within the framework of PHP 5. It deals with the development of standard WS (calculation mode) as well as WS in the database mode (using MySQL, SQLite). It compares the structured and object-oriented approach (which is preferred) to the server part of the service development.

Key words: *Web service, PHP, XML, WSDL, MySQL, SQLite.*

Introduction

The Hypertext Processor (PHP) is a widely-used general-purpose scripting language that is especially suited for the web development as it was designed to work on the web and can be embedded into HTML. It is a procedural language, in the current version 5 with object-oriented capabilities, and has a syntax similar to the C, Perl, and Java languages. PHP was first released on June 8, 1995 and was created by Rasmus Lerdorf, initially as a simple set of Perl scripts for tracking accesses to his online resume. PHP 3 was the first version that closely resembles PHP as we know it today. It was created by Andi Gutmans and Zeev Suraski in 1997 and was officially released in June 1998. The PHP version 4, released in May 2000, was a major milestone, with its explosive performance increase and many features, such as the native session support. The PHP version 5 was released in July 2004 after a long development and several pre-releases. It is mainly driven by its core, the Zend Engine 2.0, with a new object model and dozens of other new features, first of all the advanced support of XML and web services (SOAP extension) (Gilmore 2005). Among other new features there are exceptions handling in try-catch mode, advanced string processing and the support of the SQLite open source database engine.

Several features of PHP’s are advantageous for the development of web services (WS). The first one is its object-oriented programming capabilities, namely in the version 5. It also allows SOAP (Simple Object Access Protocol) and XML-RPC (Remote Procedure Call) toolkits split into a group of classes, each supporting parts of the entire WS transaction. Another advantage of PHP is its XML support. The Expat parser is bundled with PHP, providing the SAX (Simple API for XML) capability out of the box. For the expanded XML functionality, there are several PHP extensions, such as the domxml extension (Document Object Model), the xslt (eXtensible Stylesheet Language Transformation) extension as well as the experimental extension for XML-RPC and SOAP (Ayala et al. 2002).

PHP was, along with Perl, one of the frontrunners in the server-side programming a long time before any JSP/Servlet or ASP (Active Server Page) technology came to be (Rubio 2004). It is often the language of choice for those using the Apache's Web server, which runs almost 70% of sites on the Web. Due to its pervasiveness, it seems obvious that it should support the most recent standards, such as SOAP, which are also adopted by major technology vendors. In this article, we will describe how Web services can be implemented in the PHP 5 environment.

Web services

The Web services are software applications identified by URI (Uniform Resource Identifier), whose interfaces and interconnections can be defined, described, and searched for as XML artifacts. They support a direct interaction with other software applications by means of mes¬sages written in the XML language and transported by Internet protocols (Buranský 2003).

The Web services, a stack of emerging standards that describe the service-oriented and component-based application architecture, are built on the service-oriented architecture (SOA) (Samtani 2002).

[1] *doc. Ing. Pavel Horovčák,* CSc.,Department of Applied Informatics and Process Control,Technical University of Košice,,Košice, Slovak Republic, Pavel.Horovcak@tuke.sk
(Recenzovaná a revidovaná verzia dodaná 3. 6. 2006)

A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols (Champion et al. 2002).

The WSDL (Web Services Description Language) is an XML format for describing network services as a set of endpoints operating on messages containing either a document-oriented or procedure-oriented information (Christensen et al. 2001). 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 a description of endpoints and their messages regardless of what message formats or network protocols are used to communicate. However, the only binding in this document describes how to use WSDL in conjunction with SOAP 1.1, HTTP (HyperText Transfer Protocol) GET/POST, and MIME (Multipurpose Internet Mail Extensions).

PHP and web services

The PHP up to the version 4 does not have a standard SOAP or XML-RPC support. This is the reason why there are several different implementations supporting the WS creation. First of all, it is NuSOAP, a collection of PHP classes that allow users to send and receive SOAP messages over HTTP. It is an open source, licensed under the GNU LGPL, written by Dietrich Ayala (Ayala 2005). It has been used as the core of several WS toolkits for PHP, including PEAR-SOAP and Active State software's simple WS API project. It is written in pure PHP and operates under the PHP version 4. A very helpful extension is the CURL extension, described as a Client URL Library. The CURL allows to communicate via different protocols such as HTTP, HTTPS, FTP, telnet, and LDAP (Lightweight Directory Access Protocol). Other implementation is WS initiative named Simple WS API (SWSAPI), a standard method for scripting languages to access Web services described with the WSDL. To get started with SWSAPI, see ASPN QuickStart (2005). The Komodo (ASPN 2005) is a professional IDE for open source languages, providing a powerful workspace for editing, debugging and testing applications. It supports Perl, PHP, Python, Tcl, XSLT, and numerous other languages, and runs under Linux and Microsoft Windows. Komodo features: the ability to automatically generate Perl Web services clients from WSDL files; Web services management including the bookmark management and automatic creation of Web services documentation; and AutoCompletion and CallTips for Web services objects.

There are two other methods of consuming and producing WS - XML-RPC (David 2004) and REST (Trachtenberg 2003). The Remote Procedure Calls are used to establish and facilitate transactions between two remote systems. To enable PHP XML-RPC functionality, you must download the XML-RPC toolkit, which includes xmlrpc.inc (the base class library) and xmlrpcs.inc (the server class library). The REST (which stands for the "Representational State Transfer"), is a simpler approach than XML-RPC or SOAP, using standard HTTP methods such as GET, POST and PUT to send and retrieve XML data. You can then use tools like PHP DOM, SAX, or even XSL to do the parsing.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "DTD/xhtml1-strict.dtd">
<html>
<head>
  <meta http-equiv="author" content="horovcak" />
  <meta http-equiv="Content-Type" content="text/html;
charset=Windows-1250" />
  <title>Temp client2</title>
</head>
<body>
<h3>Temperature client 2 (2 functions)</h3>
<?php
  $client = new SoapClient("teplota2.wsdl");
  $result = $client->getTemp(0);
  print("We generated temp ". $result. " deg C<BR>");
  if (is_soap_fault($result)) {
    trigger_error("SOAP Fault: (faultcode: {$result-
>faultcode}, faultstring: {$result->faultstring})",
E_ERROR);
  }
  print $client->VolumSurf(3)."<br>";
  print $client->VolumSurf(1)."<br>";
?>
</body>
</html>
```

Fig. 1. Illustration of WS client code.

```
<?php
class LocTemp {
  public function getTemp($symbol)  {
     $temp = rand(0,40);
     return $temp;
  }
  public function VolumSurf($size){
     $surf=6*$size*$size;
     $volume=$size*$size*$size;
     return "Input=".$size." Surf=".$surf." Vol=".$volume;
  }
}
$server = new SoapServer("teplota.wsdl");
$server->setClass("LocTemp "); //object approach
//$server->addFunction("getTemp");   //functional
//$server->addFunction("VolumSurf");
$server->handle();
?>
```

Fig. 2. Illustration of WS server code.

Web services in PHP5

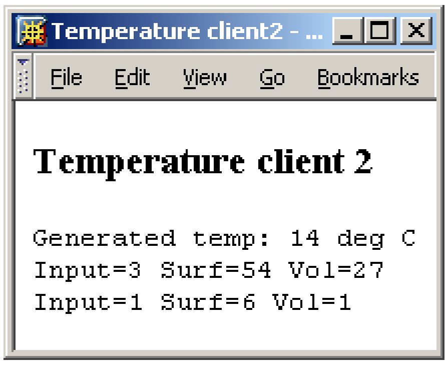
The Version PHP 5 reacts to the SOAP implementation popularity at other producers by its insertion between standard extensions. The SOAP functionality is available by a dynamic extension in the php.ini file (extension=php_soap.dll). In this way, PHP 5 provides simple development possibilities of the server as well as the client WS parts. At the same time it provides a number of functions needed for a correct WS functionality. In the WS development process is desirable to set soap.wsdl_cache_enabled=0 in the initialization file php.ini in the [soap] section (; enables or disables the WSDL caching feature). For the WS routine operation purposes the standard setting is soap.wsdl_cache_enabled=1. The fundamental prerequisite of the client and server part collaboration is a correctly configured WSDL file. This file consists of four sections which specify parameters and return values of particular server functions (<message>), port type together with particular functions names (<portType>, <operation>), particular functions binding to the port type (<binding>), and the service name with the server part‘s service URL specification (<service>). A repeatedly cited example of such a file is Temperature Service (2001). We show a few examples of typical tasks solution in the PHP5 environment, which can be divided into computational type tasks and database type tasks. In the server side realization, the object oriented approach is preferred, although it is possible to work on the standard function level. Using the object oriented approach, all methods of a given class are included into the server functionality by means of only one method (SoapServer->setClass()). It exports all methods from the specified class. Using the functional approach, it is necessary for each function to be included separately into the server functionality by means of a corresponding method (SoapServer->addFunction()). A list of all accessible server functions on the client side provides a very useful standard function SoapClient->__getFunctions(), whose return value is an array containing the names of these functions together with the names and types of its parameters.

WS of computational type

The server side of WS contains one or more functions. These functions can have zero, one or more input parameters of various types (e.g. string, integer, float). The simplest way is to use only one output parameter in each function, which is mostly of the string type. Into this output string we can easily include all needed output parameters

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='LocTemp'
  targetNamespace='urn:hp_php5_soap'
  xmlns:ph='urn:hp_php5_soap'
  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/'>

<!-- Parameters and return values -->
<!-- Function getTemp -->
<message name='getTempRequest'>
  <part name='symbol' type='xsd:string'/>
</message>
<message name='getTempResponse'>
  <part name='Result' type='xsd:int'/>
</message>
<!-- Function VolumSurf -->
<message name='VolumSurfRequest'>
  <part name='size' type='xsd:int'/>
</message>
<message name='VolumSurfResponse'>
  <part name='Result' type='xsd:string'/>
</message>

<portType name='TempTypPortu'>
  <operation name='getTemp'>
    <input message='ph:getTempRequest'/>
    <output message='ph:getTempResponse'/>
  </operation>
  <operation name='VolumSurf'>
    <input message='ph:VolumSurfRequest'/>
    <output message='ph:VolumSurfResponse'/>
  </operation>
</portType>

<binding name='TempBinding' type='ph:TempTypPortu'>
  <soap:binding style='rpc'
    transport='http://schemas.xmlsoap.org/soap/http'/>
  <operation name='getTemp'>
    <soap:operation soapAction='urn:hp_php5_soap#getTemp'/>
    <input>
      <soap:body use='encoded' namespace='urn:hp_php5_soap'
       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </input>
    <output>
      <soap:body use='encoded' namespace='urn:hp_php5_soap'
       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </output>
  </operation>
  <operation name='VolumSurf'>
    <soap:operation soapAction='urn:hp_php5_soap#VolumSurf'/>
    <input>
      <soap:body use='encoded' namespace='urn:hp_php5_soap'
       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </input>
    <output>
      <soap:body use='encoded' namespace='urn:hp_php5_soap'
       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </output>
  </operation>
</binding>

<service name='LocTemp'>
  <port name='TempPort' binding='TempBinding'>
    <soap:address

location='http://localhost/php_ws/ph_f/server2.php'/>
  </port>
        </service>
        </definitions>
```

Fig. 3. View of wsdl file.



Temperature client 2

```
Generated temp: 14 deg C
Input=3 Surf=54 Vol=27
Input=1 Surf=6 Vol=1
```

Fig. 5. WS client output

```
function VolumSurf($size){
   $surf=6*$size;
   $vol=$size*$size;
   $vysledok["size"]=$size;
   $vysledok["surf"]=$surf;
   $vysledok["volume"]=$vol;
   return print_r($vysledok,true);
 }

print_r($client->VolumSurf(3));

Array
(
   [size] => 3
   [surf] => 18
   [volume] => 9
)
```

Fig. 4. Working with array in WS

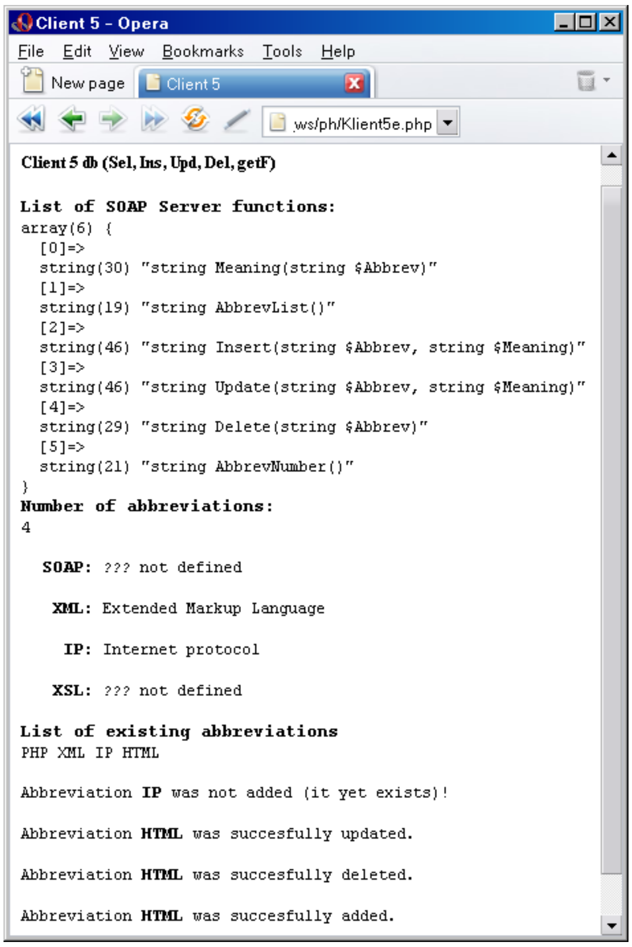
along with a suitable commentary. The PHP 5 allows the function return value in the form of an array (with string indexes), which must be processed both on the server side as well as on the client side by means of the standard function print_r(). The above-mentioned function enables an array dump (to the viewing screen or to the string). An example of WS server with two functions is presented on Fig. 1. An example of implementation of the WS client side is presented in Fig. 2. An illustration of wsdl files ensuring the binding between the server and the client side, is for our demonstration in Fig. 3. The result of the WS client side is illustrated in Fig. 4. The principle and a demonstration of how to work with arrays on the WS server side as well as on its client side, together with an example of the corresponding client's side output, is illustrated in Fig. 5. On both WS sides the function print_r is used simultaneously.

WS of database type with a selection function

Our demo WS operates over the table of acronyms with two columns, containing an acronym and its meaning. All access parameters to the database are gathered in an auxiliary file, which is a part of each server's function of service. The input of the WS is the acronym shape, and its output is the string containing a given acronym and its meaning. If the given acronym is not defined in the table, the output contains the string "??? not defined". The basic WS function can be extended by other (informative) functions, such as for example the total number of acronyms defined in the table (output is the string indicative of this number) or a function returning the list of all acronyms defined in the table. The WS of database type with the selection function usually have one (or no) input parameter. In the case of more complicated structures, relations between tables or selection conditions, the WS can also have more input parameters.

WS of database type with a record function

WS enables a new record addition, record modification, and a deletion of table records. The number of input parameters for the above-mentioned functions depends on the client authentication method (access rights) and the data model structure (and primary keys form) over which the WS operates. Illustrative WSs are developed using the already mentioned



Client 5 db (Sel, Ins, Upd, Del, getF)

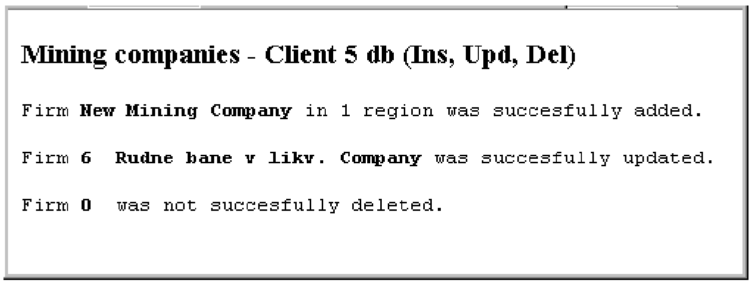
```
List of SOAP Server functions:
array(6) {
  [0]=>
  string(30) "string Meaning(string $Abbrev)"
  [1]=>
  string(19) "string AbbrevList()"
  [2]=>
  string(46) "string Insert(string $Abbrev, string $Meaning)"
  [3]=>
  string(46) "string Update(string $Abbrev, string $Meaning)"
  [4]=>
  string(29) "string Delete(string $Abbrev)"
  [5]=>
  string(21) "string AbbrevNumber()"
}
Number of abbreviations:
4

   SOAP: ??? not defined

    XML: Extended Markup Language

     IP: Internet protocol

    XSL: ??? not defined

List of existing abbreviations
PHP XML IP HTML

Abbreviation IP was not added (it yet exists)!

Abbreviation HTML was succesfully updated.

Abbreviation HTML was succesfully deleted.

Abbreviation HTML was succesfully added.
```

Fig. 6. Output of complex database WS.

acronym table. A demonstration of the complex WS output is presented in Fig. 6. The service contains a list of all server functions, the number of acronyms in the database table, the meaning of given a acronym, list of all acronyms as well as the operation of acronym addition, modification, and the deletion.

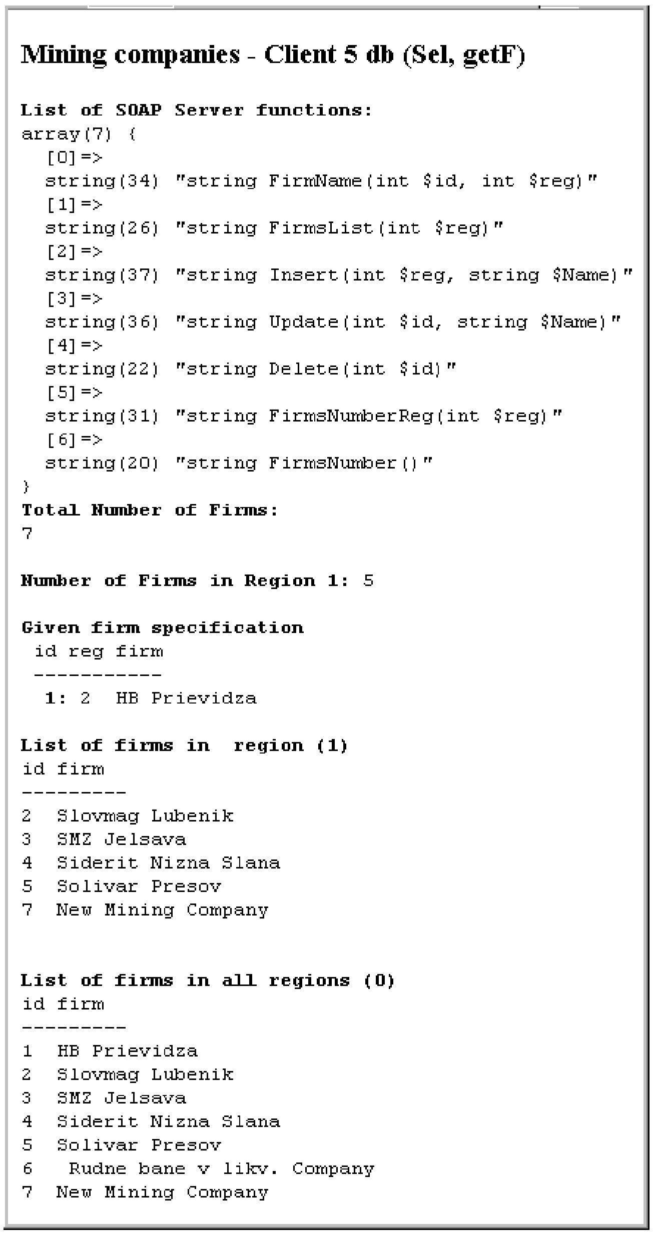
WS and SQLite

The PHP5 is coming with a new database engine SQLite, which is a major contribution. SQLite is in accordance with SQL92. It saves the database into one file only, supports transactions, is free for Windows as well as for Unix. In PHP5.0, SQLite is just a part of the environment. From PHP 5.1 upward it is necessary to ensure its operation by means of a dynamic extension in the php.ini file (extension=php_sqlite.dll), but the extension=php_pdo.dll must also be included. The development of WS using SQLite is very much like that using MySQL. Among the differences is e.g. a different way of database open (it replaces two MySQL functions: connect and select_db) and it does not use the free_result operation. In most standard operations it suffices to replace the mysql prefix with sqlite prefix in the function name. The support of SQLite in PHP has been developed for a considerably shorter time (and apparently is not yet completed) than that of MySQL, but nevertheless SQLite brings several new and effective possibilities (e.g. output formats) and functions. The database can also be placed in the memory (quickness), it supports operations with binary data as well as user defined functions (UDF) creation and even the creation of aggregating UDF for the use in SQL statements. SQLite represents a fast and efficient backend with minimal maintenance costs. Its important advantage is the possibility of transferring (in the form of a file) of the entire database between various computers. Therefore, the SQLite seems to be an appropriate candidate primarily in the database WS development stage with a possibility of subsequent trouble-free transfer to e.g. MySQL. All the above-mentioned examples of database WS operate also in the SQLite environment.



```
Mining companies - Client 5 db (Ins, Upd, Del)

Firm New Mining Company in 1 region was succesfully added.

Firm 6  Rudne bane v likv. Company was succesfully updated.

Firm 0  was not succesfully deleted.
```

Fig. 7. The output of companies evidence – records.



```
Mining companies - Client 5 db (Sel, getF)

List of SOAP Server functions:
array(7) {
  [0]=>
  string(34) "string FirmName(int $id, int $reg)"
  [1]=>
  string(26) "string FirmsList(int $reg)"
  [2]=>
  string(37) "string Insert(int $reg, string $Name)"
  [3]=>
  string(36) "string Update(int $id, string $Name)"
  [4]=>
  string(22) "string Delete(int $id)"
  [5]=>
  string(31) "string FirmsNumberReg(int $reg)"
  [6]=>
  string(20) "string FirmsNumber()"
}
Total Number of Firms:
7

Number of Firms in Region 1: 5

Given firm specification
 id reg firm
 -----------
  1: 2  HB Prievidza

List of firms in  region (1)
id firm
---------
2  Slovmag Lubenik
3  SMZ Jelsava
4  Siderit Nizna Slana
5  Solivar Presov
7  New Mining Company


List of firms in all regions (0)
id firm
---------
1  HB Prievidza
2  Slovmag Lubenik
3  SMZ Jelsava
4  Siderit Nizna Slana
5  Solivar Presov
6   Rudne bane v likv. Company
7  New Mining Company
```

Fig. 8. The output of companies evidence – selections.

WS application in the mining companies evidence

The data model of WS. To illustrate functions related to the evidence of mining companies, it is built-up the simplest data model represented by one table with three attributes id, the region and the name of a company. The id attribute represents an integer firm identification and it is a primary key of the table (auto increment). The attribute region is an integer too, whereby 1 means the East Slovak Region, 2 is the Central Slovak Region and 3 is for the West Slovak Region. The attribute value 0 stands for all the regions.

```
<?php
  $ServiceFirms = new SoapClient ("Firmy5e.wsdl");
  print $ServiceFirms->Insert(1,"New Mining Company")."<br />\n";
  $result = $ServiceFirms->Update(6," Rudne bane v likv. Company")."<br
/>\n";
  if (is_soap_fault($result)) {
    trigger_error("SOAP Fault: (faultcode: {$result->faultcode}, faultstring:
{$result->faulstring})", E_ERROR);
  }
  print $result;
  print $ServiceFirms->Delete(0)."<br />\n";
       ?>
```

Fig. 9. The part of WS client's code corresponding to output in fig. 7.

The functions of WS. The functions of mining companies evidence can be divided into two groups – the group of records and the group of selections. The group of records contain three functions – the function of the company addition, the function of the company modification and the function of the company deletion. Into the group of selection, a function of the total number of companies' x was assigned, a function of the number of companies' number in a specific region, a function of specification of a given company, a function of returning the list of all companies, a function returning the list of companies in given region, and a function providing a list of all functions of the created WS. The real operating version of each function has to be solved, handling all exceptions (by the appropriate error string). An example is the selection of a non - existing company specification or an attempt to delete the non existing company.

The application of WS. The application consists of three parts. The base part is the server part of service, which contains all the above mentioned functions, the communication to the database table, the input data check, the exceptions handling and the link to the wsdl file. All this forms conditions and requirements to its robustness and stability. The binding part is represented by the corresponding wsdl file which contains URL of the server part. The client side of service contains a link to the wsdl file and calling of one or more (eventually all) functions provided by the server side of service.

The illustration of WS. The Database table is at the beginning filled by six companies, four in the East Slovak Region and two in the Central Slovak Region. After the addition of a new company and the modification of other company (Fig. 7) the state of the evidence is illustrated in Fig. 8. For a demonstration of an exception handling (attempt to delete a non existing company) see Fig. 7. A corresponding segment of the source code of WS client's side is illustrated in fig. 9. If the function of the company specification gets an id of a non existing company, the result will be a error string in the form "??? Not defined".

The utilization of WS. The advantage of WS technology is a fact that the service's client side can be designed in other environment as the server part of service. In this way, whoever interesting in the given web service can create an access to the service in his/her conditions. By the complement and extension of the service data model, it is possible to create a companies' evidence in the needed structure. The utilization of WS in the "internal" form as a part of various applications or systems is a today's standard, the utilization of them in the "external" form becomes a part of the enterprises' information infrastructure. The current development environments by now support the possibility of WS building so that it is necessary to use them.

Conclusion

The PHP in the version 5 brings a highly-developed SOAP support, which enables a simple and effective development and exploitation of WS, too. SOAP is a lightweight protocol for the exchange of structured information in a decentralized, distributed environment, based on XML. It provides a message construction in a manner that it is possible to transfer them via various base protocols. In this way, PHP 5 joins development environments allowing the creation of the server side of a WS as well as its client side. It supports the creation of standard WSs but first of all the creation of the database type WSs. The access to the database is solved and handled on the WS server side. And since PHP is an environment with a very effective solution of the database communication, the utilization of PHP for the development of database type WS is very prospective (Jakab et al., 2002). A certain deficiency of the environment is the manual creation of the WSDL file, for which it would be better to use an XML editor. The list of provided functions (10 for the client side, 12 others) supports all standard situations encountered when working with WSs.

Acknowledgement: The contribution was created within the framework of projects KEGA 3/3084/05, KEGA 1/3126/05 (B), KEGA 1/3124/05 (L), VEGA 1/2160 /05 (K) and ICOTEL (LdV) SK/02/B/F/PP/-142261.

References

ASPN. Php Ws Quickstart. [Online] [Cited 5.12.2005] Available from Http://Aspn.Activestate.Com/Aspn/Webservices/Swsapi/Phptut.

ASPN: Php. 2005 [Online] [Cited 5.12.2005] Available From Http://Aspn.Activestate.Com/Aspn/Php/Webservices/.

Ayala, D., Browne, Ch.,Chopra, V., Sarang, P., Apshankar, K., Mcallister, T: Professional Open Source Web Services. *Wrox Press Inc 2002 Isbn: 1861007469, Pp. 523.*

Ayala, Dietrich: Nusoap - Web Services Toolkit For Php V 1.94. *[Online] 2005/08/04 [Cited 5.12.2005] Available from* <Http://Cvs.Sourceforge.Net/Viewcvs.Py/*Checkout*/Nusoap/Lib/Nusoap.Php>

Buranský, I.: XML a webové služby. Prienik do XML cez Microsoft.Net a Murphyho zákony. *Microsoft Praha 2003, 132 Str.*

Champion, M., Ferris, Ch., Newco-Mer, E., Orchard, D.: Web Services Architecture. *W3c Working Draft 14, [Online] November 2002 [Cited 5.12.2005] Available from* <Http://Www.W3.Org/Tr/2002/Wd-Ws-Arch-20021114/>

Christensen, E., Curbera, F., Mere-Dith, G., Weerawarana, S.: Web Services Description Language 1.1, [Online] *March 2001, [Cited 5.12.2005] Available from* <Http://Www.W3.Org/Tr/Wsdl>

David, Jean-Luc: Creating And Consuming Web Services With Php. Webservices.Xml.Com [Online] 2004/03/24 [Cited 5.12.2005] Available from <Http://Webservices.Xml.Com/Pub/A/Ws/2004/03/24/Phpws.Html>

Gilmore, J., G.: Velká Kniha Php5 A Mysql. *Zonerpress Brno 2005, Isbn 80-86815-20-X, 711 Str.*

Rubio, Daniel: Building Php Web Services With Pear *[Online] 2004/02/16 [Cited 5.12.2005] Available from* <Http://Webservices.Devchannel.Org/Webserviceschannel/04/02/11/1432220.Shtml?Tid=47&Tid=51&Tid=54>

Samtani, G.: Top Five Web Service Myths. Builder – Architect - Wservices *[Online] Aug. 2002 [Cited 5.12.2005] Available from* <Http://Builder.Com.Com/Article.Jhtml;Jsessionid=4vpor43osae21tqqacqsffa?Id=U00320020820gxs01.Htm&Page=2>

Jakab, F., Samuelis, L.: Internet based education of IT and computer networks in coleges of eastern Slovakia. In: *Virtual University: Proceedings of the 3rd international conference*, *Bratislava, Slovak Republic, March 17th-19th, 2002. Bratislava : STU, 2002. pp. 60-63. ISBN 80-227-1811-4.*

Temperatureservice. Wsdl *[Online] 2001 [Cited 5.12.2005] Available from* <Http://Www.Methods.Net/Sd/2001/Temperatureservice.Wsdl>.

Trachtenberg, Adam: Php Web Services Without Soap *[Online] 2003/10/30 [Cited 5.12.2005] Available from* <Http://Www.Onlamp.Com/Pub/A/Php/2003/10/30/Amazon_Rest.Html>.