

Integrated Web applications with SOAP

Girish M. Tere¹, R. R. Mudholkar², B. T. Jadhav³

¹Department of Computer Science, Shivaji University, Kolhapur, Maharashtra – 416004, India

²Department of Electronics, Shivaji University, Kolhapur, Maharashtra – 416004, India

³Department of Electronics and Computer Science, Y.C. Institute of Science, Satara, Maharashtra - 415001, India

Abstract: Recently Cloud based applications usage has increased. Web services play important role in cloud based applications. In this paper Web service is developed using PHP. PHP is server side scripting language with the power to connect to databases. Performance of Web services developed using PHP is better than those developed using .Net or J2EE technologies. A Web service is developed by building a SOAP server in PHP. A vehicle lookup service is created that takes in queries based on make, model, and year. The Web service will then query an internal database and respond appropriately. A Web-based client is also developed in PHP to communicate and query the SOAP server. Three SOAP servers in PHP are developed. Each of the three servers would be placed in three different cities or countries where a cluster of car dealerships would function. The client would then be hosted at one location where car customers would come and visit, firing search queries to find the vehicles of their needs. The client routes the query to each of the three SOAP servers, which, in turn, send results back to the client. Upon receiving each response, the client displays the search results to the user for analysis. Web services are becoming more popular because they are the perfect way to integrate several entities into one, allowing for better flow of information to management and to users. In this paper SOAP based Web service is developed which queries backend Derby database. Derby is open source light weight RDBMS having better performance than other RDBMS.

1. INTRODUCTION

Web application uses services from many servers which are developed in different platforms. Interaction between such servers and clients is possible with help of Web services. SOAP messages are exchanged between these applications. It is a standard for exchanging XML-based data via HTTP. This paper demonstrates development of PHP based Web services and uses Apache Derby database as backend. Clients can use these Web services and Web services fetches data from Derby database and returns to client.

2. SOFTWARE USED

The application is developed in Windows 7 Ultimate Operating System (64 bits). Following software is used for developing the application:

- JDK 1.7
- Database: Apaches Derby 10.10.2.0

- XAMPP Server 1.8.3-5 with PHP 5.5.15
- IBM DB2 ODBC Driver

A Web service can hardly be useful without a database. One can use a database to store information about what a particular user queried or looked at during a visit. Most existing systems of large companies have a vast use of databases and information, making the use of databases vital for a successful Web service [1,4].

Web site information is always stored in a database, and accessing such information via a Web service is just as important. The paper demonstrates how to integrate PHP Web service with a Derby database.

3. SIMPLE SOAP SERVER IN PHP

A simple SOAP server is created to understand basic PHP SOAP server capabilities [3]. This section explains development of SOAP server and how to use it.

3.1 Creating a server

A simple server takes a SOAP request and returns a response. A simple echo application is created in PHP that takes in a string and sends it back as shown in Listing 1.

Listing 1. A simple SOAP server

```
<?php
function echoo($echo)
{ return "ECHO: ".$echo;}
$server = new SoapServer(null, array('uri' =>
    "urn://teacher/res"));
$server->addFunction('echoo');
$server->handle();
?>
```

The echoo function returns the string passed to it and appends ECHO: to the front of it. The SoapServer object is created in PHP. The echoo function is added to the list of functions that the SOAP server supports. The last line calls the handle

method of the SoapServer object, allowing the server to handle the SOAP request and return a response, as defined in the echo method.

SOAP messages

Pointing a browser to SOAP server in its current status causes a fault because of the way the request is sent. The data needs to be sent as raw POST data via HTTP, as described by the fault string.

Listing 2. Pointing a browser to the SOAP server

```
<SOAP-ENV:Envelope>
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>Bad Request. Can't find
HTTP_RAW_POST_DATA</faultstring>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Live server is accessed by a SOAP client.

3.2 Creating a client: Echo form

A client allows sending data to the SOAP server using the correct expected protocol. A simple form is created to accept any string and send it to the SOAP server. Form consists of one text box and a button. Form is created using PHP code as shown in Listing 3.

Listing 3. Creating a simple form

```
<?php
$echo = $_GET['input'];
print "<h2>Echo Web Service</h2>";
print "<form action='simple_client.php'
method='GET'>";
print "<input name='input'
value='$echo'><br/>";
print "<input type='Submit' name='submit'
value='GO'>";
print "</form>";
```

Requests are sent to the SOAP server using GET or POST methods. The code first retrieves the value of input from the GET array or URL. Next, the form is created, with the action field being this same PHP script, simple_client.php, so GET requests from this form get sent to this same PHP script. There are two input tags: the text box where user will type the value to have returned from the SOAP server and the GO button. A preview of the form is shown in Figure 1.

3.2.1 Creating a client: Making the request

Once the button is clicked, the text in the text box, shown above, gets sent to the PHP script in the URL, which one can extract in the GET array. This allows us to verify that a

request was sent and process it. PHP code for handling requests and sending them to SOAP server is mentioned in Listing 4.

Listing 4. Handling requests and sending them to the SOAP server

```
print "</form>";
if($echo != ""){
$client = new SoapClient(null, array(
'location' =>
"http://localhost/soap/simple_server.php",
'uri' => "urn://teacher/req"));
$result = $client->
__soapCall("echo",array($echo));
print $result;}
?>
```

Now if \$echo has data in it, something was entered in the text box, and a request was made. This allows user to initiate the request to the SOAP server by creating the SoapClient object. The client knows where to send requests by the location in the parameters array. The uri gives the SOAP packet a namespace, which is essentially a context. Once the SoapClient is initialized, make the request to the SOAP server by calling the client's __soapCall method with two parameters: the method in the SOAP server and an array of parameters. The response sent from the SOAP server is displayed under the GO button, as shown in Figure 1.

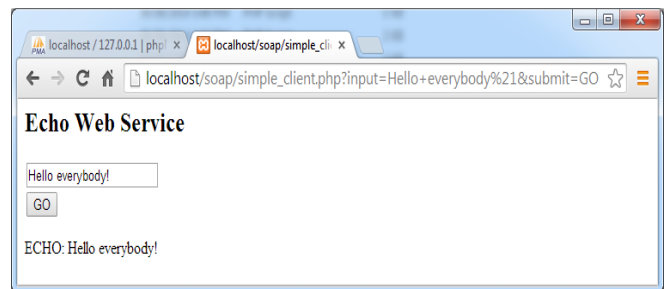


Fig. 1. Displaying response from the SOAP server

Next multiple SOAP servers are created which use Apache Derby database as backend.

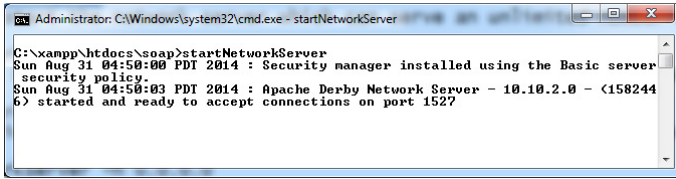
3.3 Derby: Setting up

Derby is used for this application because it's lightweight and easy to use. The application in this paper uses it to search for vehicles matching the search criteria.

3.3.1 Creating the database

Since PHP connects to Derby using the network server and ODBC [1], start the server by typing:

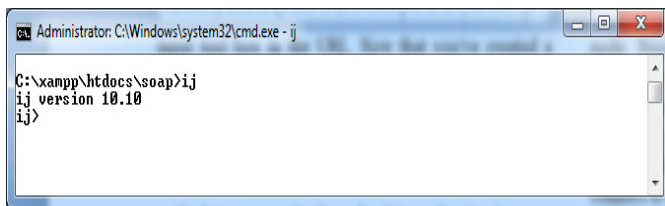
- Start Network Server
- Output is shown in Figure 2.



```
Administrator: C:\Windows\system32\cmd.exe - startNetworkServer
C:\xampp\htdocs\soap>startNetworkServer
Sun Aug 31 04:58:06 PDT 2014 : Security manager installed using the Basic server
security policy.
Sun Aug 31 04:58:03 PDT 2014 : Apache Derby Network Server - 10.10.2.0 - (158244
6) started and ready to accept connections on port 1527
```

Fig. 2. Starting Derby Database

When the application starts, it's ready to accept connections. Connect to and create a new database using the Derby ij tool as shown in Figure 3.



```
Administrator: C:\Windows\system32\cmd.exe - ij
C:\xampp\htdocs\soap>ij
ij version 10.10
ij>
```

Fig. 3. Use of ij Tool

Getting 'ij>' one can enter database commands. The ij tool allows us to create and connect to databases, as well as query them [1]. This helps developers to fine-tune and perfect search queries before implementing them in PHP application. Following command is used to create and connect to the database:

```
connect
'jdbc:derby://localhost:1527/DEALER;create=true:
user=dealer;password=dealer;';
```

Here, we have created the DEALER database with user and password being dealer. Next, appropriate PHP ODBC driver is used to connect/query to database.

3.3.2 The DEALER database

This database holds three tables -- one for each physical dealership location. Each location will have its own table in the database with vehicle data. And when queries from the SOAP client come to the SOAP server, each SOAP server will query its own database table and return the results for it. The three tables:

- vehicles_mumbai
- vehicles_delhi
- vehicles_chennai

Structure of these tables is shown in Figure 4. Each table contains the vehicles for each of three cities, which are created next.

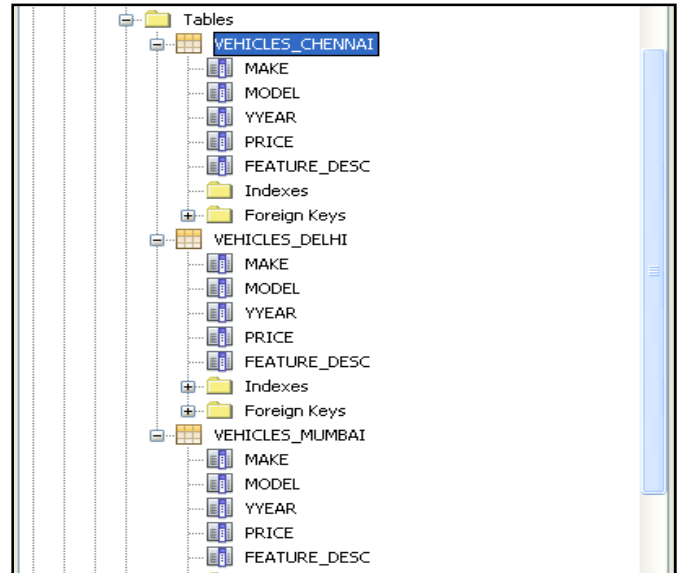


Fig. 4. Tables of DEALER database

3.3.3. Creating the vehicle tables for each location

The three tables for the dealership application are created using the ij tool. Tables are created using the SQL commands at the ij prompt, as shown in Listing 5.

Listing 5. Create the three tables

```
drop table vehicles_mumbai;
create table vehicles_mumbai (make varchar(50), model
varchar(50), yyear varchar(4), price integer, feature_desc
varchar(512));
```

```
drop table vehicles_delhi;
create table vehicles_delhi (make varchar(50), model
varchar(50), yyear varchar(4), price integer, feature_desc
varchar(512));
```

```
drop table vehicles_chennai;
create table vehicles_chennai (make varchar(50), model
varchar(50), yyear varchar(4), price integer, feature_desc
varchar(512));
```

Each table has a make, model, yyear, price, and feature_desc describing a vehicle's features. The database skeleton is complete with the creation of these tables. Next few records are inserted in these tables.

3.4 Architecting the user interface

With the database ready to go, one can begin work on the PHP application. The application user interface with a form that takes in the make, model, and year of the vehicle being searched for is shown in Figure 5.

3.4.1 The header

When a page is requested, the header appears at the top of the page, and placing those contents in a separate PHP script makes code modular and easier to read because one can simply include the header file at the top of a new script. A header.php file is created as shown in Listing 6.

Listing 6. Creating a header file

```
<?php
$COMPANY_NAME="Car shoppee for us";
print(
<html>
<title>'. $COMPANY_NAME.' </title>
<body>
<table width="650px" align="center" valign="top">
<tr><td colspan="2">
');
print(
</td></tr>
<tr><td colspan="2">
<center><h1>Welcome
'. $COMPANY_NAME.' </h1></center></td></tr>');
print(<tr><td height="100%">
');
?>
```

This sets up the HTML for the page, as well as a structure defined by the <table ...> tag and displays a greeting to visiting users. It ends by opening up a <td ...> tag.

3.4.2 The footer

The concept of a footer file is similar to the concept of a header file. With the header and footer file together, there is no need to write basic HTML tags (like <html><body>) ever again. This allows developer to focus on PHP development. A file created and called footer.php as shown in Listing 7.

Listing 7. Creating a footer file

```
</td></tr>
<tr><td align="center" colspan="2">
<font size="2px"><br>
<center>Copyright 2014, <?php print($COMPANY_NAME)
?></center>
</font>
</td></tr></table>
</body></html>
```

This file closes off the <td ...> and <table ...> tags from the header, and displays a copyright. Looking up vehicles with a form with the header and footer created, one can focus on the main content of the user interface for the application. A file called index.php in created as shown in Listing 8.

Listing 8. Creating the form used to search for vehicles

```
<?php
require('header.php');
$make = $_GET['make'];
$model = $_GET['model'];
$year = $_GET['year'];
print "<h3>Search for vehicles based on year, make or
model:</h3>";
print "<table>";
print "<form action='index.php' method='GET'>";
print "<tr><td>Make:</td><td><input name='make'
value='$make'/></td></tr>";
print "<tr><td>Model:</td><td><input name='model'
value='$model'/></td></tr>";
print "<tr><td>Year:</td><td><input name='year'
value='$year'/></td></tr>";
print "<tr><td><input type='submit' name='submit'
value='GO'/></td></tr>";
print "</form>";
print "</table>";
if($_GET['submit'] === 'GO' && ($make != "" || $model != "" ||
$year != "")){
print "<h3>Search Results:</h3>";
require('client.php');
}
require('footer.php');
?>
```

The form consists of three text input boxes: one each for make, model, and year. The code for the form is shown in Listing 8. Like the Echo form, the HTTP transfer method of this form is via GET because there will be no database or other side-effects caused by executing this query. Once the query is executing, the action field specifies that the form should send the data to this same index.php script. The first three lines of the script retrieve the data from the URL or GET array, which need to be processed next. Form is shown in Figure 5.

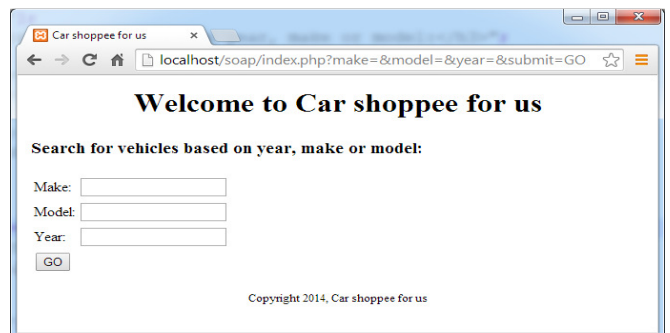


Fig. 5. Car shoppee application

3.4.3. Processing the form and calling the client

Once a request has been made by a car customer, code needs to handle that request and call client code to request responses

from each of the three SOAP servers. PHP code for handling requests from the user interface is shown in Listing 9.

Listing 9. Handling requests from the user interface

```
print "</form>";
print "</table>";
if($_GET['submit'] === 'GO' &&
($make != "" || $model != "" || $year != ""))
{

print "<h3>Search Results:</h3>";
require('client.php');
}
require('footer.php');
?>
```

The request has been made by a car customer if they GET array's submit value equals GO. Now if the button got clicked and any one or more of the text boxes was filled in, initialize a request to the SOAP servers via the client code.

3.4.4 The server

Next the SOAP server is developed. The three SOAP servers get called by the client code, and each of them returns results, as found in their own local databases. The server is developed and by connecting to and querying the Derby database desired results are obtained.

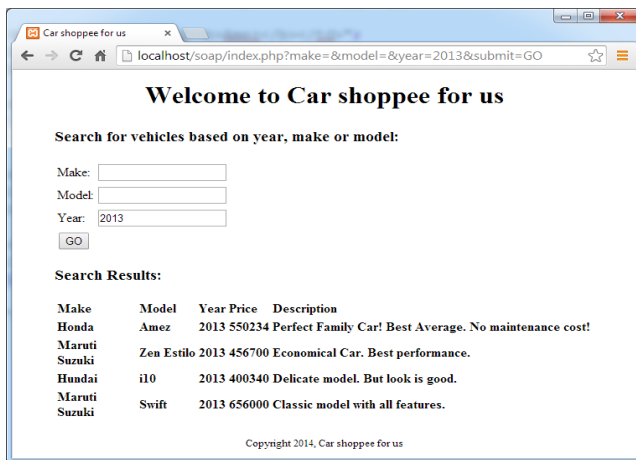


Fig. 6. Car shoppee application with search results

3.4.5 Connecting to Derby

Before the database can be queried, one needs to connect to it in PHP. A db_connect function is defined in all three sever files, as shown in Listing 10.

Listing 10. Connecting to the DEALER database

```
<?php
function db_connect($dbname='DEALER',
$username='dealer',
$password='dealer') {
$pdo = new PDO("odbc:$dbname", $username, $password);
```

```
return $pdo;
}
function vehicleLookup($make, $model, $year){
...
Output of some queries is shown in Figure 6 and Figure 7.
```

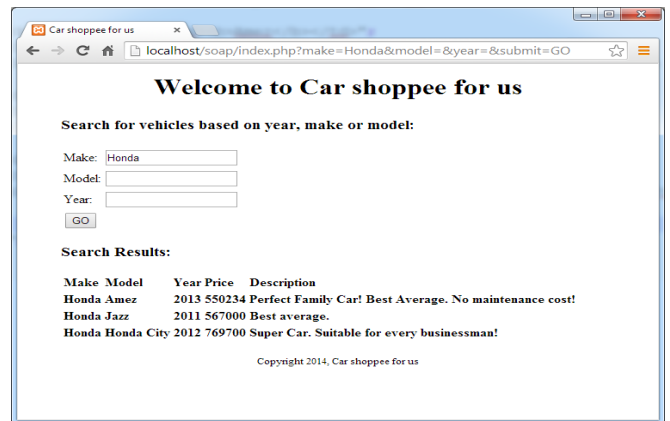


Fig. 7. Car shoppee application with search results

4. CONCLUSIONS

A SOAP server in PHP is developed. Using developed Web services heterogeneous Web applications can be developed. To develop Web services PHP programming language is good choice as it is high performance server side scripting language and has ability to connect to various databases. As PHP and Derby both are open source software the cost of developing application will reduce.

5. ACKNOWLEDGEMENTS

This work was carried at Thakur College of Science and Commerce, Mumbai. Authors thank the Management Trustees and Principal to allowing us to use various resources required to complete the research.

REFERENCES

- [1] Brett McLaughlin, PHP and MySQL: The Missing Manual, O'Reilly Media, Inc., 2012
- [2] Brian Hayes, Cloud Computing, JULY 2008, VOL. 51, NO. 7, COMMUNICATIONS OF THE ACM, pp 9-11, News Technology, DOI: 10.1145/1364782.1364786
- [3] Hasin Hayder, Object-Oriented Programming with PHP5, Packt Publishing, 2007
- [4] Lorna Jane Mitchell, PHP Web Services, O'Reilly Media, Inc., 2013
- [5] Yogesh L. Simmhan, Beth Plale, Dennis Gannon, A survey of data provenance in e-science, Newsletter ACM SIGMOD Record Homepage archive, Volume 34 Issue 3, September 2005, Pages 31-36, ACM New York, NY, USA, DOI: 10.1145/1084805.1084812