SOAP clients in PHP

Disposition

- Necessary steps when constructing a client with a WSDL
- Example client
- Necessary steps when constructing a client without a WSDL
- Example client
- Complex types
- Example client
- Error handling
- Concluding remarks

SOAP clients in PHP

Necessary steps to write a client from a WSDL.

1. Get the address of the WSDL file for the service in question
2. Write the basic client to output the PHP proxy class
3. Study the proxy to see if any method specifies complex types
4. If no methods specifies complex types goto 7
5. Copy/paste address of WSDL file to browser and study the complex types
6. Remember every complex type in your client will be a stdClass. a) Simple output means using the foreach construct.
   b) Input means using an associative array
7. You can now write the client.

SOAP clients in PHP

The example client will show how to make a client implementing the Stock Quotes web service from xmethod.net.

Step 1) Find the address for the WSDL file
- http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl

Step 2) Write the basic client for outputting the proxy

```
<?php
    $soapclient = new
SoapClient('http://services.xmethods.net/soap/urn:xmethods-delayed-
quotes.wsdl');
    echo $soapclient->__getFunctions(), “\n”;
    echo $soapclient->__getTypes(), “\n”;
?>
```

SOAP clients in PHP

Step 3) study the proxy

float getQuote(string $symbol)

In this case the proxy only has one function: getQuote.
The function takes a string as input and returns a float as output.

E.G. $quote = $soapclient->getQuote('ibm');

SOAP clients in PHP

Step 7) The Client

```
<?php
   // To be able to show debug information
   $options = array('trace' => true);
   $soapclient = new SoapClient(
   'http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl',
$options);
   try {
      echo $soapclient->getQuote1('ibm') . "\n";
   }
   catch (SoapFault $fault)
   {
      trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}\n" .
      "faultstring: {$fault->faultstring})", E_USER_ERROR);
   }
?>
```

SOAP clients in PHP

Necessary steps to write a client from scratch.

1. Fetch detailed description for the service in question
2. Study the detailed description to see if it specifies complex types
3. Find the messages describing required input and output
4. Find the names of the operations which the service provides
5. Find binding and encoding style
6. Find the namespaces for each operation
7. Find address to service
8. Create an associative array to hold input for every complex type in your client
9. Create a class for the proxy
10. Create a method in the class for each operation
11. You can now write the client.

SOAP clients in PHP

```
<definitions name="net.xmethods.services.stockquote.StockQuote"
      targetNamespace=
"http://www.themindelectric.com/wsdl/net.xmethods.services.stockquote.StockQuote/">
<message name="getQuoteResponse1">
      <part name="Result" type="xsd:float"/>
</message>
<message name="getQuoteRequest1">
      <part name="symbol" type="xsd:string"/>
</message>
<portType name="net.xmethods.services.stockquote.StockQuotePortType">
      <operation name="getQuote" parameterOrder="symbol">
            <input message="tns:getQuoteRequest1"/>
            <output message="tns:getQuoteResponse1"/>
      </operation>
</portType>
```

SOAP clients in PHP

```
<binding name="net.xmethods.services.stockquote.StockQuoteBinding"
    type="tns:net.xmethods.services.stockquote.StockQuotePortType">
    <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getQuote">
    <soap:operation soapAction="urn:xmethods-delayed-quotes#getQuote"/>
        <input>
        <soap:body use="encoded" namespace="urn:xmethods-delayed-quotes"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
        <output>
        <soap:body use="encoded" namespace="urn:xmethods-delayed-quotes"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </output>
    </operation>
</binding>
```

SOAP clients in PHP

```
<service name="net.xmethods.services.stockquote.StockQuoteService">
    <documentation>
        net.xmethods.services.stockquote.StockQuote web service
    </documentation>
    <port name="net.xmethods.services.stockquote.StockQuotePort"
        binding="tns:net.xmethods.services.stockquote.StockQuoteBinding">
        <soap:address location="http://64.124.140.30:9090/soap"/>
    </port>
</service>
</definitions>
```

SOAP clients in PHP

Step 9) Create the class for proxy

```
<?php
    class StockQuoteProxy {

        private $soapclient;
        private $debug;

        public function __construct($debug = false) {
            $this->soapclient =
                new SoapClient(null, array(
                    'location' => 'http://64.124.140.30:9090/soap',
                    'uri' => 'urn:xmethods-delayed-quotes'),
                    'trace' => $debug);
            $this->debug = $debug;
        }
```

SOAP clients in PHP

Step 10) Create a method for each operation

```
    public function getQuote($symbol) {
        $arguments = array(new SoapParam("$symbol", 'symbol'));
        $options = array(
            'uri' => 'urn:xmethods-delayed-quotes',
            'soapaction' => 'urn:xmethods-delayed-quotes#getQuote');
        $result = $this->soapclient->__soapCall('getQuote', $arguments,
$options);
        if ($this->debug) {
            echo $this->soapclient->__getLastRequest(), "\n";
            echo $this->soapclient->__getLastResponse(), "\n";
        }
        return $result;
    }
}
```

SOAP clients in PHP

Step 11) Write the client

```
<?php
    require_once ('StockQuoteProxy.php');

    $client = new StockQuoteProxy(/* true */);
    echo $client->getQuote('ibm'), "\n";
?>
```

SOAP clients in PHP

A complex type

```
<complexType name=”test”>
    <all>
        <element name="id" type="xsd:int" />
        <element name="name" type="xsd:string" />
    </all>
</complexType>
```

PHP

```
return
class test {
    public $id = 1;
    public $name  = 'Some Name';
};
```

```
internal
$test = array(
        'id' => 1;
        'name'  => 'Some Name';
);
```

SOAP clients in PHP

Example client receiving a complex type

```
<?php
    $soapclient = new SoapClient(
        'http://greg.froh.ca/fun/random_bushism/soap/?wsdl');
    // Invoke the method getRandomBushism and print
    try {
        $result = $soapclient->getRandomBushism();
        foreach ($result as $output)
            echo $output, "\n";
    }
    catch (SoapFault $fault)
    {
        trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}\n" .
        "faultstring: {$fault->faultstring})", E_USER_ERROR);
    }
?>
```

SOAP clients in PHP

Error handling

- The SOAP extension provide try catch for error handling.

```
try {
    $service->invoke();
}
catch (SoapFault $sf)
{
    trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}\n" .
    "faultstring: {$fault->faultstring})", E_USER_ERROR);
}
```

- The SOAP extension also provide debug information

```
    $service = new SoapClient(wsdl, array('trace' => true));
    echo $service->__getLastRequestHeaders();
    echo $service->__getLastRequest();
    echo $service->__getLastResponseHeaders();
    echo $service->__getLastResponse();
```

SOAP clients in PHP

Concluding remarks

- Use hash arrays to create structs
- If output is arrays or structs they will be returned as PHP objects
- Always provide debugging capabilities to your clients by using , $instance->__getLastRequest(), $instance->__getLastResponse()
- Always put call inside a try catch block to activate error handling and take appropriate actions, should errors arise.