PHP web services with the NuSOAP library

Fulvio Corno, Dario Bonino

e-lite Research Group
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino - Italy
`http://elite.polito.it`

v. 2.1, April 2, 2009

Outline

Goals

- Create and use Web Services using the RPC messaging model
- Being able to create a Web Service in PHP (server-side)
- Being able to call an existing Web Service from PHP code (client-side)
- Hiding, as much as possible, the creation of SOAP messages, and their decoding
- Avoid the difficulty of creating/editing WSDL files

Potential Solutions

- Php comes with a “standard” `SOAP` library
  - `http://www.php.net/soap`
  - tutorial at `http://devzone.zend.com/node/view/id/689`
- The PEAR library has a SOAP module
  - `http://pear.php.net/package/SOAP`
  - probably the most updated, but not well documented
- The NuSOAP library (we will use this)
  - `http://sourceforge.net/projects/nusoap/`

Limitations

- All considered libraries are easy to use when the input(s) and output of the web service are simple types (numbers, strings)
- Complex types are difficult to “teach” to the libraries, and they are not fully supported
- For practical reasons, we will try to use only `xsd:string` types

Outline

NuSOAP characteristics

- Written in PHP (no new modules to install or configure)
- Simple object-oriented interface
- May work with or without WSDL files
- May automatically generate a WSDL file for the service

Installation

- Download from `http://sourceforge.net/projects/nusoap/`
- Uncompress the file, and copy the `lib` directory under your PHP project
- In your PHP files (client or server), import the needed classes
  - `require_once('lib/nusoap.php');`

Outline

SOAP Server

Minimal SOAP server

```
require_once('lib/nusoap.php') ;

$server = new nusoap_server; // Create server instance

$server->register( 'myFunction' ) ;

function myFunction($parameters) {
  . . .
  return $result;
}

// Use the request to (try to) invoke the service
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA)
    ? $HTTP_RAW_POST_DATA : '';
$server->service($HTTP_RAW_POST_DATA);
```

SOAP Server

Step 1: create the server

```
// Create the server instance
$server = new nusoap_server;
```

- `$server` is the special variable with the SOAP server functionality

SOAP Server

Step 2: identify the service function

```
$server->register( 'myFunction' ) ;
```

- registers a new function (SOAP “operation”) in the same web service

SOAP Server

Step 3: implement the service function

```
function myFunction($parameters) {
  . . .
  return $result;
}
```

- function containing the implementation of the web service
- receives the SOAP input(s) directly as funtion parameter(s)
- its return value is automatically packaged as the SOAP return value

SOAP Server

Step 4: execute the web service

```
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA)
    ? $HTTP_RAW_POST_DATA : '';
$server->service($HTTP_RAW_POST_DATA);
```

- `$HTTP_RAW_POST_DATA` should contain the XML SOAP request
- `$server->service` analyzes the XML, calls the function, and creates the XML response
- the actual web service has already been called (http) to get the current page

Outline

SOAP Client

Minimal SOAP client

```
require_once('lib/nusoap.php');

// Create the client instance
$client = new nusoap_client('http://localhost/ws.php',
    false); // false: no WSDL

// Call the SOAP method
$result = $client->call('myFunction',
    array('param' => 'abc'));

print_r($result) ;
```

SOAP Client

Step 1: create the client

```
$client = new nusoap_client(
    'http://localhost/ws.php',
    false ); // false: no WSDL
```

- `$client` is the special variable with the SOAP client functionality
- the first parameter is the URL of the web service endpoint
- the second parameter is **`false`**: no WSDL is used

SOAP Client

Step 2: call the web service

```
$result = $client->call(
    'myFunction',
    array('param' => 'abc') );
```

- the first parameter is the name of the called function
- the second parameter is an array with the list of SOAP inputs
- the array contains a list of parameter name => parameter value couples

SOAP Client

Step 3: use the result

```
print_r($result) ;
```

- the result is available in a PHP variable
- usual PHP type conversions apply

Outline

Using WDSL

- the WSDL file should describe the web service, and in particular the name(s) and type(s) of input and output parameters
- in some cases, we already have a WSDL description file
- in other cases, we want the server to automatically provide a WSDL file describing its services

Server support for WDSL

Support WSDL generation

```
$server->configureWSDL('demows',
    'http://example.org/demo') ;
```

- tell the NuSOAP server that we want to support WSDL generation
- parameters: name of the service, namespace URL

Server support for WDSL

Declare input/output types

```
$server->register( 'myFunction',
    array("param"=>"xsd:string"), // inputs
    array("result"=>"xsd:string"), // outputs
    'http://example.org/demo' // element namespace
) ;
```

- for each function, declare types of input and output parameters
- name => type arrays
- type in XSD syntax

Self-documenting web service

- With the above server declarations, the web page `http://localhost/ws.php` may be called in 3 ways:
  1. with an http POST, by providing an XML body: the normal Web Service call
  2. with an http GET, with a normal browser: shows a human-readable description of the service
  3. with an http GET and a `?wsdl` parameter: generates the WSDL on-the-fly and lets you download it

Client support for WDSL

Call using WSDL information

```
$client = new nusoap_client(
    'http://localhost/ws.php?wsdl',
     true);
```

- the client asks for the WSDL file (that is generate on-the-fly)
- the second parameter set to true specifies that the first one is the address of the WSDL file (and not of the endpoint)

Outline

Error Checking - Client

Error types

error can either happen during the client creation or during the evaluation of call results

fault can happen during the remote service call

Service creation

```
//build the nu-soap client
$client = new nusoap_client(
  'http://localhost/ws.php?wsdl',true);
//check errors
$err = $client->getError();
if($err)
  //write the error and do not perform the call
  echo "<strong>Error:</strong>".$err;
```

Error Checking - Client

Service call

```
//do the call
$callResult = $client->call("myFunction",
  array("param"=>"abc"));

//check fault
if($client->fault)
{
    echo "<strong>Fault:</strong>".
      print_r($callResult);
}
```

Error Checking - Client

Call results

```
//check fault
if($client->fault) {
  echo "<strong>Fault:</strong>".
  print_r($callResult);
} else {
  //check error
  $err = $client->getError();
  if($err)
  {
    //write the error
    echo "<strong>Error:</strong>".$err;
  }
  else { ... } //result ok
}
```

Outline

Complex Types

NuSOAP supports the definition of Complex Types, i.e., of complex data structures that may be exchanged by web service calls. Complex Types may be:

- Composed of Simple `xsd` types
- Composed of other Complex Types
- May include (unbound) sequences or arrays of Complex or Simple types
- Declared as PHP **`array`**s or `struct`s (associative arrays)

Complex Types: structures

Structure types

```
$server->wsdl->addComplexType(
  'Theater',
  'complexType',
  'struct',
  'sequence',
  '',
  array(
    'idTheater' => array(
      'name' => 'idTheater', 'type' => 'xsd:int'),
    'Name' => array(
      'name' => 'Name', 'type' => 'xsd:string'),
    'Address' => array(
     'name' => 'Address', 'type' => 'xsd:string')
  )
);
```

Generated Schema for structures

Automatically generated WSDL

```
<xsd:complexType name="Theater">
 <xsd:sequence>
  <xsd:element name="idTheater" type="xsd:int"/>
  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="Address" type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>
```

Complex Types: arrays

Array types

```
$server->wsdl->addComplexType(
  'Theaters',
  'complexType',
  'array',
  '',
  'SOAP-ENC:Array',
  array(),
  array(
    array(
      'ref'=>'SOAP-ENC:arrayType',
      'wsdl:arrayType'=>'tns:Theater[]')
    ),
    'tns:Theater'
);
```

Generated Schema for arrays

Automatically generated WSDL

```
<xsd:complexType name="Theaters">
  <xsd:complexContent>
    <xsd:restriction base="SOAP-ENC:Array">
      <xsd:attribute ref="SOAP-ENC:arrayType"
       wsdl:arrayType="tns:Theater[]"/>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Using Complex Types

Service registration

```
$server->register("getTheaters",
  array("param"=>"xsd:string"), // inputs
  array("result"=>"tns:Theaters"), // outputs
  'http://example.org/demo'); // element
```

Further information

- Software and function documentation: `http://sourceforge.net/projects/nusoap/`
- Tutorial introduction: `http://www.scottnichol.com/soap.htm` (warning: uses an old version of the library)
- Slides: `http://www.slideshare.net/sanil/develop-webservice-in-php` and `http://www.slideshare.net/harit/web-services-in-php-1094228`

Outline

License