Footprints Web Services Integration

Version 0.1

Written By: Mariellen Mihopoulos
Wednesday, 4th April 2007
Background

Footprints is a dot Net application exposing select functionality through SOAP based webservices which allow donation to Footprints charities. For dotNet utilisation of these webservices the procedure is quite simple with the functionality that is inbuilt to the framework. For other application servers the concept is similar even though the implementation may differ slightly.

This document covers integration solutions at a high level and in a detailed manner for a PHP implementation. From the PHP walk through and code supplied, most developers will be able to abstract their own solution for their particular application server.

The actual IA for the donation functionality can be implemented in various ways – the donation amount and charity can be decided by the application or the affiliate, or the customer can be given a list to choose from and could also enter their own donation amount. This can be discussed with the Surecan team about what may work best.

Methodology

1. implement SOAP libraries in your technology language
2. generate proxy code by connecting to the footprints wsdl
3. create a proxy for connecting to the webservice either using the proxy code generated or by editing it if required
4. gather all the required data for a call to the webservice method
5. make the call to the webservice method using the proxy object
6. catch any errors and deal with the returned response in the appropriate manner
7. log donations into a table with the appropriate flags for “published” and “deleted” fields:- if the response code from the payment call was Success, the published value should be set to 1, otherwise it should be set to 0. Deleted field should be defaulted to 0.
8. update your application to modify your donations record for cancelled transactions so that the “deleted” field is 1 for cancelled policies
9. liaise with Footprints regarding implementation of watchdog applications for unsuccessful transaction reporting back to the Footprints server and for updating “cancelled policies” transactions in the Footprints database.

Requirements

1. Access to the footprints webservice URL from your webserver.
2. Code for requesting and receiving SOAP data over HTTP
3. A login and password for a user with permission to connect to the footprints webservice and access to at least one footprints project
4. Integration of some client side code into your purchase path pages for allowing customers to select to donate money to a Footprints charity
5. Ability for Footprints server to poll your donations table for unsuccessful transactions or cancelled transactions.

PHP Integration Walkthrough

Summary

TID is running on PHP 4.x. After investigating the potential libraries available that could be used without re-compiling PHP we decided to try nuSOAP and then PEAR::SOAP. nuSOAP’s XML
parses was not up to scratch so we tried PEAR::SOAP. This required us to copy all the PEAR php files and all the SOAP php files into the our docroot. A problem we experienced with the SOAP XML parsing was fixed with a small modification to their Base class. The generated proxy code was not functional so it was implemented as a separate class and modified.

Method

1. Copy the PEAR libraries and the SOAP libraries into the web docroot

2. Create a wsdl using the footprints web service

   ```php
   ```

3. Generate and view the proxy code and copy the generated code to new class file (see footprints_proxy_class.php in the Appendix B). These will be the classes you utilise instead of the generated code as the generated code has errors.

   ```php
   echo $soapwsdl->generateProxyCode();
   ```

4. The currentItems method should be modified to read as follows.

   ```php
   function &GetCurrentItems($username, $password, $status, $isLive)
   {
       $GetCurrentItems =& new SOAP_Value('{http://tempuri.org/}GetCurrentItems',
       false, array('username' => $username, 'password' => $password, 'status' => $status,
       'isLive' => $isLive));
       $result = $this->call('GetCurrentItems',
       $v = array('GetCurrentItems' => $GetCurrentItems),
       $parameters = array('namespace' => 'http://tempuri.org/',
       'soapaction' => 'http://tempuri.org/GetCurrentItems',
       'style' => 'document',
       'use' => 'literal',
       'trace' => '1'));
       return $result;
   }
   ```

5. Modify the SOAP Base.php _serializeValue so that the name of the array items is flattened in the XML string to be the key from the key value pairs passed through in the array (instead of the name of the item being fixed as “item”). Look for the string “Serialize each array element” in the code.

   ```php
   // Serialize each array element.
   ```
```php
$ar_size = count($value);
foreach ($value as $k => $array_val) {
    if ($this->_isSoapValue($array_val)) {
        $array_type = $array_val->type;
        $array_types[$array_type] = 1;
        $array_type_ns = $array_val->type_namespace;
        $xmlout_value .= $array_val->serialize($this);
    } else {
        $array_type = $this->_getType($array_val);
        $array_types[$array_type] = 1;
        if ($k == '') {
            $k = "item";
        }
        $xmlout_value .= $this->_serializeValue($array_val, $k, $array_type, $this->_section5 ? null : $elName);
    }
}
```

If you don’t change this, the generated SOAP envelope will contain data that looks like the following, which is not what the server is expecting (see the WSDL for the correct Request and expected Response format).

Incorrect envelope data

```xml
<ns4:GetCurrentItems>
    <ns4:item>tidnz</ns4:item>
    <ns4:item>t1dnz</ns4:item>
    <ns4:item>10</ns4:item>
    <ns4:isLive>true</ns4:isLive>
</ns4:GetCurrentItems>
```

6. Create the donations table for keeping your records of the transactions as supplied in Appendix D.

7. Test using code as supplied in Appendix C. This will attempt to create a new Soap client, make a call to the "GetCurrentItems" method to return a list of available projects, allow the user to select from this list of projects, choose a donation amount and on submission will make the payment call using dummy data and then log an entry into your donations table. Confirm the script returns a list of projects and the payment call is successful and that your donations table has recorded the data correctly.

8. If the test code works, implement accordingly into your application.

   a. NOTE: PEAR::SOAP returns different objects when the result is an array of size 1 and an array of size more than 1. Code to accommodate for this is detailed in Appendix E.

9. Modify your application to update the donation table when a transaction is cancelled and test the functionality
// now update any donations for this cancelled policy
$sql = "update donation set deleted = '1', lastmodified = now() where policyid = " . $id . "";
if ($db->Query($sql)) {
    $errmsg .= "<br/>Donation Record Updated.";
}

10. Contact footprints who will confirm your transactions have gone through successfully. Detail your donations table to Footprints and they can then implement the watchdog applications for synchronising failed transactions or cancelled purchases back to the Footprints server.

11. Test, test test ☺

DotNet Integration Walkthrough

Appendix A - Definitions

Footprints Service Description


What are Web Services?

Open standard (XML, SOAP, etc.) based Web applications that interact with other web applications for the purpose of exchanging data. Initially used for the exchange of data on large private enterprise networks, web services are evolving to include transactions over the public Internet.

What is PEAR?

PEAR is the PHP Extension and Application Repository. It is a framework and distribution system for reusable PHP components.

http://pear.php.net/

What is SOAP?

SOAP (Simple Object Access Protocol) is a simple XML-based protocol to let applications exchange information

What is PEAR::SOAP?

A PHP implementation of SOAP protocol and services.

http://pear.php.net/package/SOAP/redirected
<?
include('SOAP/Client.php');

class FootprintsWebServiceGenerated extends SOAP_Client
{
    function FootprintsWebServiceGenerated($path = 'http://dev.footprints.org.au/WebService/SecurityWebService.asmx')
    {
        $this->SOAP_Client($path, 0);
    }
    function &GetCurrentSuppliers($donatorusername, $donatorpwd)
    {
        $GetCurrentSuppliers =& new SOAP_Value('{http://tempuri.org/}GetCurrentSuppliers', false,
        $v = array('donatorusername' => $donatorusername, 'donatorpwd' => $donatorpwd));
        $result = $this->call('GetCurrentSuppliers',
            $v = array('GetCurrentSuppliers' => $GetCurrentSuppliers),
            array('namespace' => 'http://tempuri.org/',
                'soapaction' => 'http://tempuri.org/GetCurrentSuppliers',
                'style' => 'document',
                'use' => 'literal'));
        return $result;
    }
    function &GetCurrentItems($username, $password, $status, $isLive)
    {
        $GetCurrentItems =& new SOAP_Value('{http://tempuri.org/}GetCurrentItems', false,
            array('username' => $username, 'password' => $password, 'status' => $status, 'isLive' => $isLive));
        $result = $this->call('GetCurrentItems',
            $v = array('GetCurrentItems' => $GetCurrentItems),
            $parameters = array('namespace' => 'http://tempuri.org/',
                'soapaction' => 'http://tempuri.org/GetCurrentItems',
                'style' => 'document',
                'use' => 'literal',
                'trace' => '1'));
        return $result;
    }
    function &DoPayment($username, $password, $itemId, $amount, $currencyCode, $name, $email, $extensionnumber)
    {
        $DoPayment =& new SOAP_Value('{http://tempuri.org/}DoPayment', false,
            $v = array('username' => $username, 'password' => $password, 'itemId' => $itemId,
                'amount' => $amount, 'currencyCode' => $currencyCode, 'name' => $name, 'email' => $email,
                'extensionnumber' => $extensionnumber));
        $result = $this->call('DoPayment',
            $v = array('DoPayment' => $DoPayment),
            array('namespace' => 'http://tempuri.org/',
                'soapaction' => 'http://tempuri.org/DoPayment',
                'style' => 'document',
                'use' => 'literal',
                'trace' => '1'));
    }
}
function &CancelPayment($username, $password, $extensionnumber)
{
    $CancelPayment =& new SOAP_Value('{http://tempuri.org/}CancelPayment', false, $v = array('username' => $username, 'password' => $password, 'extensionnumber' => $extensionnumber));
    $result = $this->call('CancelPayment', $v = array('CancelPayment' => $CancelPayment), array('namespace' => 'http://tempuri.org/', 'soapaction' => 'http://tempuri.org/CancelPayment', 'style' => 'document', 'use' => 'literal'));
    return $result;
}

Appendix C – Test client code

<?
require("includes/class_footprints_proxy.php");  // check this path is correct

$fpwsg = new FootprintsWebServiceGenerated($footprints_WebService_URL);
$currentItems = array();

$posted = cleanPost($_POST['posted']);
if ($posted == 1)
{
    $donationAmount = cleanPost($_POST['donationAmount']);
    $charityId = cleanPost($_POST['footprintsCharityId']);

    // set dummy name, email, currencycode and policynumber
    $name = "Your name";
    $email = "your@email.address";
    $policynumber = "TIDAUSHOL07123456";
    $currencycode = "AUD";

    // make the payment!
    $payment = $fpwsg->DoPayment($footprints_WebService_username,$footprints_WebService_password, $charityId, $donationAmount, $currencycode, $name, $email, $policynumber);

    print $payment->ResponseCode;
    if (PEAR::isError($payment))
    {
}
$published = 0;
} else if ($payment->ResponseCode == "Success")
{
  // donation successful
  $published = "1";
} else {
  // donation unsuccessful for some reason
  $published = "0";
}

// record transaction in donations table
$sql = "insert into donation (footprintsprojectid, charityprojectname, " .
  "policynumber, policyid, " .
  "amount, currencycode, " .
  "paymentgateway, created, " .
  "lastmodified, published, deleted, sitename) ";
$mask = "%s", "%s", "%s", "%s", %s, %s, %s, %s", now(), now(), "%s", "%s", "%s" ";
$sql.= " VALUES (";
$sql.= sprintf($mask,
  sqlNumber($policy_bundle["footprints_charityprojectid"]),
  sqlString($policy_bundle["footprints_charityproject"]),
  sqlString($policy_bundle["policy_number"]), sqlNumber($policy_bundle["policy_id"]),
  sqlString($policy_bundle["footprints_donationamount"])),
  sqlString($desccurrencystring),
  sqlString($descpaymentgateway), sqlNumber($published), "0",
  sqlString($descsitename));
$sql .= ");
$db->Query($sql);
} else {
  $currentItems = $fpwsg->GetCurrentItems($footprints_WebService_username, $footprints_WebService_password, "10", "true");
  ?>
<form method="post">
<input type="hidden" name="posted" value="1">
<table border="0">
  <tr>
    <?
    if (sizeof($currentItems) > 0)
    {
      foreach ($currentItems as $p)
      {
        ?>
        <tr>
          <td>
            <?
            if ($p->ProjectImageThumURL != "")
            {
              ?>
            </td>
          </tr>
        <?
      }
    ?>
  </table>
  <![CDATA[
  $currentItems = $fpwsg->GetCurrentItems($footprints_WebService_username, $footprints_WebService_password, "10", "true");
  ?>
<form method="post">
<input type="hidden" name="posted" value="1">
<table border="0">
  <tr>
    <?
    if (sizeof($currentItems) > 0)
    {
      foreach ($currentItems as $p)
      {
        ?>
        <tr>
          <td>
            <?
            if ($p->ProjectImageThumURL != "")
            {
              ?>
            </td>
          </tr>
        <?
      }
    ?>
if ($p->CharityWebsite != "") {
    ?><a href="<? print $p->ProjectURL; ?>" target="_blank"><a href="<? print $p->ProjectURL; ?>" target="_blank">?
    <img src="<? print $p->ProjectImageThumURL; ?>' width="20" border="0">
    </a><?
} else {
    ?>

<input type="radio" name="footprintsCharityId" value="<? print $p->ProjectId; ?>" target="_blank"><a href="<? print $p->ProjectURL; ?>" target="_blank">?
    $p->ProjectTitle;
    if ($p->CharityWebsite != "") {
        ?><a href="<? print $p->ProjectURL; ?>" target="_blank">?
    }
} else {
    ?></a><?

<h5><? print $p->CountryName; ?></h5>
</td>
</tr>?
</table>

<br />
donate: AUD $&nbsp;
<select name="donationAmount" id="ddlAmount">
<option value="0">0</option>
Appendix D – Donations table schema

```sql
create table donations (  
donationid int(11) primary key auto_increment,  
footprintsprojectid int(11) not null,  
# required  
charityprojectname varchar(100) not null,  
# required  
policynumber varchar(32) not null,  
# required  
policyid int(11) not null,  
# required  
amount decimal(10,2) not null,  
# required  
currencycode varchar(5) not null,  
paymentgateway varchar(60),  
created datetime default 'NOW()' not null,  
lastmodified datetime not null,  
published int(1) null,  
# required  
deleted int(1) not null,  
# required  
sitename varchar(128)  
)
```

Appendix E – dealing with Arrays of size 1 with PEAR::SOAP

```php
if (! PEAR::isError($currentItems))
{
   if (is_array($currentItems))
   {
      $footprints_charity = $currentItems[rand(0, sizeof($currentItems) - 1)];
   } else {
      $footprints_charity = $currentItems->ProjectInfo;
   }
} else {
   print "<!-- PEAR ERROR: ". $currentItems->message . " -->";  
}
```