O'REILLY®

8TH ANNUAL OSCON™ OPEN SOURCE CONVENTION

Dirty Secrets of PHP 5's ext/soap

Adam Trachtenberg
Senior Manager of
Platform Evangelism, eBay

JULY 24-28, 2006

eBay Developers Program

- http://developer.ebay.com
- Free to join
- Make up to 1,500,000 calls/day
- Make money by remixing eBay to help our 203 million buyers and sellers use our marketplace in new and interesting ways.

SOAP Is A Dirty Business

- SOAP 101
- ext/soap 101
- Debugging
- Headers (Authentication)
- Redefining Endpoints
- Intercepting the PHP Method (Before)
- Intercepting the SOAP Request (After)
- Class Mapping
- Attributes
- Everything Else
- Final Thoughts

SOAP 101

- Language-neutral versions of serialize() and unserialize()
- In theory, specification allows many generalizations
- In reality, data structures formatted to and from XML sent and received via HTTP(S) POST
- Quite helpful to use a specialized extension to manage the translation and transportation, say ext/soap.
- Thus abstracting out those particular details
- But abstractions are leaky
- And the specification is difficult to understand
- This requires **you**, brave coder, to stare and stare at the messy XML that is SOAP to debug the call.

The Messy XML That is SOAP

```
POST /wsapi?callname=GetUser&siteid=0&version=467&appid=ADAMMACCABM7F714274W4IE2M27882&Routing=default HTTP/1.1
Host: api.ebay.com
Connection: Keep-Alive
User-Agent: PHP-SOAP/5.2.0-dev
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 1539

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:ebay:apis:eBLBaseComponents"><SOAP-
ENV:Header><ns1:RequesterCredentials><ns1:eBayAuthToken>AgAAAA**AQAAAA**aAAAAA**/Zl1QQ**nY+sHZ2PrBmdj6wVnY
+sEZ2PrA2dj6wJkYepAZCKoA6dj6x9nY+seQ**4wQAAA**OAMAAA**vWmMixB3j6PEtkJfIRd7YWKP3fhydfNhsRquW7zh6WT5csX7HSq0kjXWg5Rp2nG/
8Uglv6ihMk5LHYzWpmyYUu9Bht5LhlZLF1pvtkFaK3267tmeqcyDU+8mX1eNp+lUslsFJhLaNyZFSyeJSMeDFm243j9RIlLmcNjx8/
rSc0Fsen7A7z3M2K63ya4KJU34tPvyfSSHWCanqgOHTx4dYqTqrFL6sZvORa5tUIt4JPKkKT5jOoWVOmKqyOW
+GEsjcBuIlswhGsjubzlXFmhuszDZzMxbKxIMqCCflXoR1CxXInm4g3o+bFJZNah2/
tadvQXw1nJo2NNaDTc1o9wdE4brr3MXnC8A35E8vitkqdtFsVx4JFs2aXuYAwl/rrM//KT2L7gDG4+V2ZMb+UkvB2eEWJYXOKsFdPJXdOzWoFFe69fxetfKS8
+8MDYVBS2i+TdqwiuBh0NCaqG9jIjT0wE3PYsfSxW1WzTLvk/h7oat8JOxY+71kyrvQirywJCcsG98Wpe/bQGI25zK2YIupTakxTHBCf0R/prjy3ryjoQ/7GaRFK/
80mUYmzG4h5jCW/nQ2ilFC5YJF0+OwhKCW81KpK89r1kQoVJ2ecyD/
UGCwLor8P9Wa8dHOgUdThpLM5LImR1NjPtEmT9V9ysIUn2ALqT5bksPAplDhalgJBLrjmemXNdg9vci0pYxMKVcpxab0KcF03cvyS0YhbLyMWvvc2J5ZeUlL3Oi0O3s
3KUyWdTN6QfeH8jVzodxaRhLPQKC6TDQMGEZud3ED7/cCS/WnKP0Wt2x59hj531HWr8b6iflZF2NvA**</
ns1:eBayAuthToken><ns1:Credentials><ns1:AppId>ADAMMACCABM7F714274W4IE2M27882</
ns1:AppId><ns1:DevId>H47LA7CN65J1E1DRYRDJI61V2238F2</ns1:DevId><ns1:AuthCert>A3G23U95I29$1EEOC9848-OU2VI4BA</ns1:AuthCert></
ns1:Credentials></ns1:RequesterCredentials></SOAP-ENV:Header><SOAP-ENV:Body><ns1:GetUserRequest><ns1:Version>467</
ns1:Version></ns1:GetUserRequest></SOAP-ENV:Body></SOAP-ENV:Envelope>

HTTP/1.1 200 OK
Date: Tue, 25 Jul 2006 04:15:30 GMT
Server: WebSphere Application Server/4.0
Content-Type: text/xml; charset=utf-8
X-EBAY-API-SERVER-NAME: ___dXUucmd3ajUyNCw0NikyNSgxNys3MzA/Pjc7NQ==
Content-Language: en
X-Cache: MISS from propel.sjc.ebay.com  MISS from propel.sjc.ebay.com
```

Never Fear

- I have seen the enemy and it is SOAP
- But I have learned all its dirty secrets
- I will show you how to use ext/soap to tame even the most unruly of SOAP servers

ext/soap 101: Amazon E-commerce WS

```
$client = new SoapClient(
  "http://soap.amazon.com/schemas2AmazonWebServices.wsdl");
$params = array(
  'keyword' => 'adam trachtenberg', /* ... */ );
$result = $client->KeywordSearchRequest($params);
foreach ($result->Details as $product) {
    echo $product->ProductName . "\n";
}
```

PHP Cookbook
Upgrading to PHP 5

WSDL and XML Schema

- WSDL file describes the service
- Specifies methods, input arguments, return values, types
- This file can get quite large
- The eBay WSDL is 2.7 megs. That's 75k+ lines.
- XML Schema describes the data structures
- As a result, variables are typed
- Understanding these specifications can be quite useful

Debugging

- Wrap your code inside a try/catch block
- Find the XML of a working request
- Enable the trace option
- Use the __getLast*() methods
- Read the WSDL

Debugging: Amazon E-commerce WS

```
try {
  $opts = array('trace' => true);
  $client = new SoapClient(
  "http://soap.amazon.com/schemas2/AmazonWebServices.wsdl",
  $opts);
  $params = array(
    'keyword' => 'adam trachtenberg', /* ... */ );
  $result = $client->KeywordSearchRequest($params);
  foreach ($result->Details as $product) {
    echo $product->ProductName . "\n";
  }
} catch (SOAPFault $f) {
  print $f;
}
```

Debugging: getLast*() Methods

```
try {
/* ... */
} catch (SOAPFault $f) {
  print $f;
}
print "Request: \n".
       $client->__getLastRequestHeaders() ."\n";
print "Request: \n".
       $client->__getLastRequest() ."\n";
print "Response: \n".
       $client->__getLastResponseHeaders()."\n";
print "Response: \n".
       $client->__getLastResponse()."\n";
```

Debugging: getLast*() Methods

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope ...>
  <SOAP-ENV:Body>
    <ns1:KeywordSearchRequest>
      <KeywordSearchRequest xsi:type="ns1:KeywordRequest">
      <keyword xsi:type="xsd:string">adam trachtenberg</
keyword>
        ...
    </ns1:KeywordSearchRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Headers

- SOAP headers go in SOAP's <Header> block
- Similar in spirit to HTTP headers
- Most often used with authentication credentials
- For all requests: __setSOAPHeaders()
- For individual requests: __soapCall()
- **Must create headers by hand using SOAPHeader**

Headers: Google AdWords

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope ...
  xmlns:ns1="https://adwords.google.com/api/adwords/v4">
  <SOAP-ENV:Header>
    <ns1:email>adam@trachtenberg.com</ns1:email>
    <ns1:password>Secret Password</ns1:password>
    <ns1:useragent>OSCON AdWords Demo</ns1:useragent>
    <ns1:token>Secret Token</ns1:token>
  </SOAP-ENV:Header>
...
```

Headers: Google AdWords

```
$ns        = 'https://adwords.google.com/api/adwords/v4';
$email     = new SOAPHeader($ns, 'email',        $email);
$password  = new SOAPHeader($ns, 'password',   $password);
$useragent = new SOAPHeader($ns, 'useragent', $useragent);
$token     = new SOAPHeader($ns, 'token',        $token);
$headers   = array($email, $password, $useragent, $token);

$client    = new SOAPClient($wsdl);
$client->__setSOAPHeaders($headers);
```

Headers: eBay

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope ...
  xmlns:ns1=" urn:ebay:apis:eBLBaseComponents">
  <SOAP-ENV:Header>
    <ns1:RequesterCredentials>
      <ns1:eBayAuthToken>My Token</ns1:eBayAuthToken>
    </ns1:RequesterCredentials>
  </SOAP-ENV:Header>
...
```

Headers: eBay

```
$eBayAuthToken = 'My Token';
$eBayAuth = array('eBayAuthToken' =>
  new SOAPVar($eBayAuthToken, XSD_STRING, null, null,
                                          null, $ns);
$headerBody = new SOAPVar($eBayAuth, SOAP_ENC_OBJECT);
$header = new SOAPHeader($ns, 'RequesterCredentials',
                                  $headerBody);
$client->__soapCall($method, $args, array(),
                        array($header));
```

Redefining Endpoints

- Change the request URL specified in WSDL
- Staging versus Production
- Per-call data (routing, authentication, performance)
- For all requests: __construct(), __setLocation()
- For individual requests: __soapCall()

```
new SOAPClient($wsdl, array('location' => $location));
$this->__setLocation($location);
$this->__soapCall($function, $args,
   array('location' => $location), $this->headers);
```

Intercepting the PHP Method (Before)

- Alter the endpoint location, add SOAP headers, etc. on a per-call basis without exposing the kludgy __soapCall() syntax.
- Subclass SOAPClient and implement a __call() method

Intercepting the PHP Method (Before)

```
class eBaySOAP extends SoapClient {
  public function __call($function, $args) {
    $query_string = http_build_query(array(
      'callname' => $function, /* ... */);
    $location = "{$this->location}?{$query_string}";
    return $this->__soapCall($function, $args,
      array('location' => $location), $this->headers);
  }
}

$eBay = new eBaySOAP($wsdl);
$eBay->GetUser();
```

Intercepting the SOAP Request (After)

- Alter the SOAP XML document to implement manipulations ext/soap doesn't support or to wrangle your XML into a state compatible with a non-compliant SOAP Server
- Subclass SOAPClient and create a __doRequest() method

Intercepting the SOAP Request (After)

```
class SalesforceSOAP extends SOAPClient {
  public function __doRequest($request, $location,
                              $action, $version) {
    $dom = new DOMDocument('1.0');
    $dom->loadXML($request);
    // ... manipulate XML ...
    $request = $dom->saveXML();
    return parent::__doRequest($request, $location,
                               $action , $version);
  }
}
```

Attributes

- When the return data has attributes, ext/soap maps the attribute name to object properties.
- The text inside the element gets mapped to “_”

```
<StartPrice currencyID=”EUR”>85</StartPrice>
```

```
[StartPrice] => stdClass Object
  (
    [_] => 85
    [currencyID] => EUR
  )
```

Class Mapping

- By default, XML Schema complexTypes are mapped to StdObjects.
- You can make ext/soap map them to specific PHP classes
- Lets you simplify usability by implementing iterators, stringifications, ArrayAccess, etc.
- No ability (currently) to call a method, but plans to add support for __sleep() and __wakeUp().
- Use the classmap option in the constructor

Class Mapping

```
class eBayAmountType {
  public function __toString() {
    return (string) $this->Fee->_;
  }
}
```

```
$classmap = array('AmountType' => 'eBayAmountType');
$options = array('classmap' => $classmap);
$eBay = new SOAPClient($wsdl, $options);
```

Class Mapping: eBay Fees

```
<Fees>
  <Fee>
    <Name>AuctionLengthFee</Name>
    <Fee currencyID="USD">1.0</Fee>
  </Fee>
  <Fee>
   <Name>BoldFee</Name>
   <Fee currencyID="USD">0.0</Fee>
  </Fee>
  ...
</Fees>
```

Class Mapping

```
class eBayFeesType implements ArrayAccess {
  public function offsetGet($offset) {
    foreach ($this->Fee as $value) {
      if ($value->Name == $offset) {
        return $value;
      }
    }
  }
  // other interface methods
}

echo "Listing fee: ", $result->Fees['ListingFee'], "\n";
```

Everything Else

- Setting a Custom XML Schema Type: xsi:type attribute
- Google AdWords

```
<ns1:job xsi:type="ns1:CustomReportJob">...
```

```
$data = array(‘name’ => ‘OSCON Test’, /* ... */);
$job = new SOAPVar($data, SOAP_ENC_OBJECT,
                   'CustomReportJob', $ns);
$response = $client->scheduleReportJob(
                array('job' => $job));
```

Everything Else

- Enabling compressed requests using gzip:

```
$options = array('compression' => SOAP_COMPRESSION_ACCEPT |
   SOAP_COMPRESSION_GZIP | 9); // 9 is the gzip level
$client = new SoapClient($wsdl, $options);
```

- XML Security: Ask Rob Richards. :)

http://www.cdatazone.org/index.php?/archives/9-WSSE-and-extsoap.html

More Information

- http://www.php.net/soap
- *Upgrading to PHP 5* (OSCON bookstore today)
- *PHP Cookbook, 2ed* (Pre-order for August or September)