


Friday, July 20. 2007


PHP OpenID 2.0 library for PEAR: Preview available from subversion

So after a few days throwing around code in an IDE, a small amount of swearing, and an overdose of caffeine, I have committed initial OpenID PHP5 code in it's shiny new PEARified form to subversion. What this means is that after a few more days of completion to work out major kinks, the PEAR-Dev mailing list will be notified of a new OpenID Consumer proposal 

. A Server proposal will follow at a later date, i.e. when the dull ache in my forearms subsides.

<http://svn.astrumfutura.org/pear/trunk/OpenID/>

Feel free to checkout a copy from subversion for a preview. Just don't shoot me if it's not currently working; it needs at least two more days before it's feature set is covering the main use cases. At present, I can only verify it works when authenticating against an OpenID 2.0 Server (i.e. no OpenID 1.1 servers just yet - try again tomorrow). The problem with specification coding is that after you've written around 95% of the source code required, nothing works - it's always the last 5% that binds the entire thing into a functioning unit.

All I'm announcing here is that a public repository is open, and the code is for the first time available under an open source New BSD License. Up until this point there's been no public code for anyone to read through. Now you can check it out, break it, complain to me, and explain how much of a steaming dogpile it all is 

For those losing track of the plot, this is the exact same body of code (only in a much advanced state of refactoring) which was originally targeted for the Zend Framework since February 2007. When I was ready to publish proposals, I requested final feedback from the ZF mailing lists and received a reply within a few hours announcing that Zend had an in-house OpenID project and would be publishing their proposal during the week.

I was pretty critical of both the convenient timing of Zend's urge to make an OpenID proposal, and the lack of awareness on anyone's behalf of my pre-existing effort, and admittedly still find it mystifying. At the time I pulled my proposal completely, and well, here it is one month later heading to PEAR. Perhaps one day this code, and some of the related proposals I put back in place after some discussion with Dmitry Stogov, will still end up in the framework. I haven't ruled that out, and I look forward to discussing OpenID with Dmitry in the future.

Now, where would code be without examples? I'll be working on some documentation eventually, but here's the quickie version to get anyone so inclined to use unstable, days from working perfectly, code started off.

Your first step should be to install three PEAR packages. These are previous PEAR proposal I made to support OpenID operations, and you can download the current packages from <http://padraic.astrumfutura.com/pear/>. To install, open up the command line and run the following PEAR command:

```
pear install Crypt_DiffieHellman-0.1.0a3.tgz
```


Do the same for the Crypt_HMAC2 and Services_Yadis packages. Services_Yadis should also get PEAR to install the Validate and HTTP_Request packages automatically since they are dependencies.

Next checkout the OpenID source code. I haven't put this into a PEAR package yet since it still needs a spot

of work before being ready for review, so you manually checkout/export using a Subversion client and copy to a location on your include_path.

Here's a quick test authentication script to start you off. I'm 99.5% sure this currently works - so long as your OpenID is from an OpenID 2.0 totting Provider (1.1 authentication will be in place by tomorrow). MyOpenid.com for example supports 2.0 authentication. Note, if the response result matches the OpenID::OPENID_RESPONSE_SUCCESS constant value then authentication has succeeded.

```
require_once 'OpenID/Consumer.php';
require_once 'OpenID/Store/File.php';
$store = new OpenID_Store_File(dirname(<u>_FILE_</u>)); // use current working dir to cache
association data used for verifying response signatures
$test = new OpenID_Consumer($store);
if (isset($_GET) && !empty($_GET)) {
    $result = $test->finish($_GET);
    if ($result->getResult() !== OpenID::OPENID_RESPONSE_SUCCESS) {
        exit('Paddy\'s library screwed up or the Server did! Result was:' . $result->getResult()); // "parse
error" is often a library screw-up ; )
    }
    echo 'Authenticated with User OpenID: ', $result->get('openid.claimed_id'), ' The OpenID Provider
knows you as ', $result->get('openid.identity'), '<br/>';
    exit();
}
OpenID::setVersion(2.);
$authRedirect = $test->start('padraic.myopenid.com');
$authRedirect->redirect('http://localhost/path/to/this/file', 'http://localhost'); // last param only needs to
be the base URI scheme and domain name
```

Try not to break anything, ok? 

Edit: OpenID 1.1 Authentication should now also work in most cases. The Sreg issue for OpenID 2.0 was resolved. Sreg for 1.1 is likely non-functional - this has a lower priority, and the reason is simple that Sreg has two versions, and I'm using the most recent one. Needs some logic to switch between the two.

Posted by Pádraic Brady in Openid and Yadis, PHP General, PHP Security at 23:06

Hey Paddy,


What about changing the OpenID_Response_Authorization class to use __get() instead of all of the accessor methods? It would be much cleaner from my perspective to have:

```
$result->openid['identity'];
$result->openid['sreg']['email'];
$result->result;
// etc., etc.
```

Add in private accessors and __get() is just a simple Strategy implementation:

```
public function __get($key)
{
$method = "get_{$key";
```

```
// do check for unknowns
return $this->$method();
}
```

Looks good though. I'm glad I didn't have to write it 

FYI: Your hint on the geshi plug-in isn't correct, it closes with /geshi, and that plugin still has htmlentities() called on it so all of my ">" became encoded making the code unintelligible. Anonymous on Jul 21 2007, 02:21

cool, have posted about it on my site to let others know what great work you've done



Anonymous on Jul 21 2007, 05:06

Sweet! I was looking for an alternative.

I'll start using it and help test it out. Anonymous on Jul 21 2007, 07:24

Haven't looked up the spec in detail but does the response back from the Identity Server have to be in GET. If not how do I change it to POST method ? Anonymous on Jul 22 2007, 09:25

Yes, it's GET. You see the OpenID server interacts with the user and then sends them back to us (the Relying Party) using a redirect.

This, normally, should not be a problem. All responses carry a signature which is based on a binary string (the secret key) only the Server, and Consumer know about. This prevents any external party from tainting & replaying the response as the signature is unique to the content of that response. Anonymous on Jul 22 2007, 19:11