

Thursday, September 1, 2005

Flickr API Fun

I like stuff I can pick up and do something useful with in an hour or two. Perhaps my attention span is too short, but if I have to read a 300 page spec before I get to Hello World, then it's not for me. Or you would at least have to pay me a lot of money to suffer through it. I think people refer to this as "immediacy". For me I think it is mostly lazyness. If I can't figure it out in an hour, it's broken as far as I am concerned.

Flickr's REST API is not broken. You can read all about it at <http://flickr.com/services/api>.

There are links there to various wrappers for the API, but I ended up writing my own. I have a bad habit of doing that. This entry will focus on my PHP wrapper for the Flickr API. It is based on Cal's version and is compatible with it, but it expands on it and puts some PHP 5.1 features to good use. You can see it here:

http://lerdorf.com/php/flickr_api.phps

Before you get started, in case you want to follow along, go get yourself an API key at

<http://flickr.com/services/api/key.gne>

You will need two pieces of information to fully use the API. An API key and an API secret. And if you are going to do anything that requires authentication, you need to set a callback url as well. More on that later. To get your secret after applying for and getting your API key, go to

http://www.flickr.com/services/api/registered_keys.gne and click on "Edit Configuration".

Many functions in the API do not require authentication. Getting a list of someone's public photos, for example, is something anybody can do by just browsing Flickr, or by just going to this URL:

http://flickr.com/services/rest/?method=flickr.people.getPublicPhotos&user_id=56053642@N00&api_key=3aba8184848f9263b80795c95529bcd1

Guess what, you just sent a REST Web Services query.

Or, slightly cooler. A list of tags related to the tag you provide based on Flickr's clustering code.

http://flickr.com/services/rest/?method=flickr.tags.getRelated&tag=monkey&api_key=3aba8184848f9263b80795c95529bcd1

The whole point of web services is to provide data in a machine-readable way so you can do something more interesting with it. That's where the API wrapper comes in. You can of course also use Flickr's feed mechanism to do this.

But back to the API and the PHP wrapper. Getting a list of someone's public photos is done like this:

Blog Export: Rasmus' Toys Page, <http://toys.lerdorf.com/>

```
$secrets = array('api_key'=>'your_key_here','api_secret'=>'your_secret'); $flickr = new Flickr($secrets); $photos = $flickr->peopleGetPublicPhotos('56053642@N00');
```

This will give you an array of photos. Or to be precise, an array of information about the photos. Note the mapping of the method name. `$flickr->peopleGetPublicPhotos` maps to `flickr.people.getPublicPhotos` in the documentation. And the returned XML is converted to a more useful (and more memory-cacheable - I'll write something up soon on that) PHP array. The example result XML for 2 photos looks like this:

Which gets mapped to this PHP array (in `print_r` format):

```
Array (
  [page] => 1
  [pages] => 1
  [perpage] => 100
  [total] => 2
  [photos] => Array (
    [39006009] => Array (
      [id] => 39006009
      [owner] => 56053642@N00
      [secret] => f2086066d5
      [server] => 33
      [title] => IMG_7564.JPG
      [ispublic] => 1
      [isfriend] => 0
      [isfamily] => 0
    )

    [39006000] => Array (
      [id] => 39006000
      [owner] => 56053642@N00
      [secret] => 4ec57bd51f
      [server] => 28
      [title] => IMG_7551.JPG
      [ispublic] => 1
      [isfriend] => 0
      [isfamily] => 0
    )
  )
)
```

To turn a photo into a URL you can use in an IMG tag you would call the `$flickr->getPhotoURL()` method. It isn't very complex. Here is what it does:

```
function getPhotoURL($p, $size='s', $ext='jpg') { return "http://photos{$p['server']}.flickr.com/{$p['id']}{$p['secret']}{$size}.$ext"; }
```

Blog Export: Rasmus' Toys Page, <http://toys.lerdorf.com/>

The default size is the small 75x75 square thumbnail. See the URL Documentation for further info. So here is the full code to put up the first 50 thumbnails from someone's photostream:

The contents of secrets.inc is just that \$secrets array I referred to above. You can see the output of this script at flickr_demo1.php.

Flickr users are uniquely identified by a very cryptic-looking nsid. You don't see this id anywhere when you are clicking around on Flickr. But you can look up a user's nsid if you know their photo url, their user name or their email address. flickr_demo2.php shows you how to do that. Change the u= parameter in the URL to look up other users.

Playing with the non-authenticated functions of the API can get you far, but Flickr also lets you authenticate, and it will let the users using your application authenticate themselves. That lets you do a whole class of cool things that something like the RSS feed mechanism doesn't provide. For example, I wrote a Gallery to Flickr migration tool that can take my Gallery photo albums and copy the pictures to Flickr and put them in a Flickr set with the same name as the Gallery album they came from. You could also write alternative frontends for it and integrate your Flickr photos with your own web site. Or perhaps write a Gallery plugin that uses Flickr as the backend. All sorts of possibilities here.

But in order to do any sort of reading of non-public information or writing to your Flickr account via the API, you have to authenticate. Flickr uses a token-based authentication system where you make a roundtrip to flickr.com for the user to log into his flickr account and choose whether or not to grant your application the requested level of access. That means that your application never sees the user's credentials, but instead gets a token with the appropriate rights associated with it that it can then use. Each API call then includes this token, the application's key and all the arguments for whatever method you are calling and a signature using your application-specific secret across all the arguments. That means that even if someone sniffs your traffic, all they can do is replay the exact API call. They can't use it to execute arbitrary things against your account. Users can also remove an application's access later by going to <http://www.flickr.com/services/auth/list.gne>. The system is described at <http://www.flickr.com/services/api/auth.spec.html> and is worth a read if you are interested, but you don't really need to understand it. Just use the wrapper. Here is how:

This probably looks a bit cryptic. This says that if you already have a token, we simply create a \$flickr object and we are ready to go. If there is no token on the request and there is no 'frob', then redirect the user to the authentication URL which is generated by the call to \$flickr->getAuthUrl with the desired permission level as an argument. The user will then get sent back to your callback url, which you would set to this same script most likely, and on that callback we still don't have a token, but you will be called with a frob parameter. A call to \$flickr->getFrobToken turns the frob into a token. You actually get back an auth array containing not just the token but also the user's nsid, permission level for the token, username and fullname. The idea is then that you include the above

Blog Export: Rasmus' Toys Page, <http://toys.lerdorf.com/>

blurb on your pages, as flickr_auth.inc, for example and pass the token along from page to page in your web application.

Now we can write our first full little authenticated example. Not much to it. We just call `$flickr->authCheckToken` on our token to see what Flickr thinks of the token we are using.

You can see the output by clicking on flickr_demos.php and selecting flickr_demo3.

So, by having those 3 includes at the top, by the time we get control we have a fully authenticated `$flickr` object that we can start using.

So, an all-out demo. In flickr_demo4 we upload a photo:

```
$photo_id = $flickr->upload($fname,$title,$desc,$tags,$perms,0);
```

Check to see if you already have a set named "Sample Set". If you don't, create it (adding the uploaded photo at the same time):

```
$set = $flickr->photosetsCreate("Sample Set", $photo_id);
```

If you do already have that set, add the uploaded photo to it:

```
$flickr->photosetsAddPhoto($set_id, $photo_id);
```

Then we can add a note:

```
$note_id = $flickr->photosNotesAdd($photo_id,342,70,50,50,"This is Carl");
```

Get info on the photo:

```
$photo = $flickr->photosGetInfo($photo_id);
```

And get a direct URL to it:

```
$url = $flickr->getPhotoURL($photo,'m');
```

Blog Export: Rasmus' Toys Page, <http://toys.lerdorf.com/>

Have a look at the full source code
and try it by going to http://lerdorf.com/php/flickr_demos.php and clicking on demo4.

Even if you don't use Flickr, I think there are a lot of interesting things here. An interesting web service authentication mechanism, a nice and clean REST API that allows for complex operations, and some PHP 5.1 XML and stream handling if you look closely at the flickr_api code.

Posted by Rasmus in Software at 09:00