

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

Wednesday, March 2, 2005

### Buzzing the Yahoo! Search Web Services

March 22. Update: And here is the Flickr version: [flickr.progphp.com](http://flickr.progphp.com)

PHP5 has a revamped XML architecture that makes dealing with SOAP and REST Web Services extremely simple. I wrote a little demo application against Yahoo!'s new search web services. It uses the various search buzz RSS feeds to seed it or you can provide your own search terms. It then uses those terms to pull image, web and news search results which it arranges somewhat haphazardly. You can play with it at <http://buzz.progphp.com>. The orange box shows results from a news search, and when a term doesn't have enough news hits I supplement them with web search results which is shown in green to distinguish them.

Apart from a bunch of messy CSS, the application is actually quite simple. Pulling from the RSS and REST servers is trivial. Here is a one-liner to pull an RSS feed from a url:

```
$url = 'http://buzz.yahoo.com/feeds/buzzoverl.xml';  
$xml = simplexml_load_file($url);
```

The actual implementation wraps this and returns an associative array with just the title and the link, like this:

```
foreach($xml->channel->item as $item) {  
    $ret[(string)$item->title] = (string)$item->link;  
}  
return $ret;
```

For the search REST queries it isn't much harder. You build your query string:

```
$url = 'http://api.search.yahoo.com/';  
$url .= 'ImageSearchService/V1/imageSearch';  
$url .= '?query='.rawurlencode($q);  
$url .= "&appid=$appid";  
$url .= "&results=$results";  
$url .= "&type=$type";
```

I then throw a cacheing layer in front of all these so I don't hit the feeds on every request. The core of the cache layer looks like this:

```
$stream = fopen($url,'r');  
$tmpf = tempnam('/tmp','YWS');  
file_put_contents($tmpf, $stream);  
fclose($stream);  
rename($tmpf, $dest_file);
```

A straight fopen() can be used since this is a simple REST query and the result is streamed directly to a temp file which is then renamed when complete to make sure other processes never see a half-written file. Check the mtime on \$dest\_file and read it until it gets too old, then refresh it.

Although I am not using any SOAP in this particular example, it isn't much harder to pull from a SOAP service. Here is a simple example that pulls from Amazon's SOAP service (they have a REST interface as well). It caches a serialized version of the generated object based on the service index and keywords requested.

```
$amazon_index = array(
```

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

```
'DVD', 'Photo', 'Electronics', 'OfficeProducts', 'HealthPersonalCare',
'Toys', 'Baby', 'VideoGames', 'MusicTracks', 'OutdoorLiving',
'Blended', 'MusicalInstruments', 'Magazines', 'DigitalMusic',
'Jewelry', 'Video', 'Tools', 'PCHardware', 'SportingGoods',
'Classical', 'Software', 'Books', 'VHS', 'Wireless', 'Restaurants',
'Music', 'GourmetFood', 'Miscellaneous', 'Kitchen', 'WirelessAccessories',
'Merchants', 'Beauty', 'Apparel'
);

function amazon($index, $keywords, $timeout=7200) {
    $dest_file = "/tmp/aws_{$index}_".md5($keywords);
    if(file_exists($dest_file) && filemtime($dest_file) > (time()-$timeout)) {
        $result = unserialize(file_get_contents($dest_file));
    } else {
        $aws = new SoapClient('http://webservices.amazon.com/'.
            'AWSECommerceService/US/AWSECommerceService.wsdl',
            array("trace" => 1));
        $result = $aws->ItemSearch(array(
            'SubscriptionId'=>'XXXXXXXXXXXXXXXXX',
            'AssociateTag'=>'lerdorf-20',
            'Request'=>array(array('SearchIndex'=>$index,
                'Keywords'=>$keywords))
        )
        );
        $tmpf = tempnam('/tmp','YWS');
        file_put_contents($tmpf, serialize($result));
        rename($tmpf, $dest_file);
    }
    return $result;
}
```

I still much prefer the REST services out there. SOAP always reminds me of being stuck behind the guy in a hat driving a Lincoln Towncar. You eventually get to where you want to go, but the journey is painful. With REST you can just toss your query into your browser and have a look at the returned XML. SOAP starts to make more sense when the queries you are sending get more complex than just tossing a couple of keywords to a search service and setting a couple of flags. But don't even try to read the SOAP spec. If you managed to fight your way through that spec already, try the new WSDL 2.0 Draft Spec. This is the sort of stuff that makes my brain hurt.

And yes, I know the thumbnails don't jump to the front in IE. IE's z-index handling on position: absolute elements is braindead. So use Firefox or Safari or some other browser with decent CSS support. Also, you'll need to let the cookie through. It's just a javascript cookie with your window dimensions so I'll know how big to make the oval. And no, it isn't really meant to be useful. Just a bit of fun visual candy.

Posted by Rasmus in Software at 22:38