

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

Sunday, November 6, 2005

More fun with the Yahoo! Maps API

I spent a few hours with the Javascript-Flash Yahoo! Maps API today. The result is here:

<http://lerdorf.com/map>

Have a look at the source. There is a source link at the top-right on the page. It is about 90 lines of HTML and Javascript and virtually no server-side scripting.

I usually throw PHP code at all sorts of problems, but in this case I could do pretty much everything I wanted to directly from Javascript via the excellent API. The initial zooming from way out is a bit distracting and doesn't work too well, but I wanted to play with that a bit. Once you are zoomed in and searching for things, the search is updated as you move around the map and everything is event-based so the page doesn't need to be redrawn from scratch.

Both input fields are free-form. You can put a city name, a zip code or a full address in the Location field and in the What field you put what you are looking for. There is a special hack that checks for something like 4* and the filters the results to only show you those entries that were rated 4* or higher on Yahoo! Local. You can of course also just put something like "pizza" or "mexican" in that field.

Most of the magic here is done by 2 things. The LocalSearchOverlay and the event handling. Note how Map.EVENT_MOVE and Map.EVENT_ZOOM_END are registered events in the onInitialize() function. When you scroll the map or zoom it the onOverlayInit function will get called and the LocalSearchOverlay will be recalculated for the new map coordinates. Same thing happens when you change something in the input fields. The updateMap() function is called which will center the map at the new location and update the LocalSearchOverlay appropriately. There are a few more Javascript tricks here and there in it, like updating the link at the top so you always have a way to grab a link to your current search and send it to someone, but other than that there really isn't all that much to figure out here. Once you understand which events happen when and which methods are available where, you can do some really powerful things with this. It is all documented here:

<http://developer.yahoo.net/maps/flash/V2/flashReference.html>

and people are discussing it here:

<http://groups.yahoo.com/group/yws-maps/messages>

In my last entry I showed how to parse a geocoded XML file and put markers for each entry on the map. Someone asked me how I would do the using the JS-DHTML API instead of the JS-Flash API. It's a whole lot harder in DHTML, but it works. You can see it here:

<http://lerdorf.com/php/ymap/dquakes.php>

And I talk a bit about it here:

<http://groups.yahoo.com/group/yws-maps/message/612>

Posted by Rasmus in Software at 17:09

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

Thursday, November 3, 2005

GeoCool!

Web 2.0 and the programmable web that I and others have been talking about for a while has mostly been vapourware so far. There are a few generic components that are useful, but it is somewhat limited what you can do with them. And yes, you may consider this a somewhat biased view, but I think Yahoo!'s new geocoding platform is a huge step in the right direction.

There is of course the fancy new maps.yahoo.com/beta site which is fun, but as far as I am concerned the killer app here is the geocoding platform that drives this. And it is completely accessible for anyone to use. It's also a sane API that anybody can figure out in minutes. Here are a few tips for using this API from PHP 5.

Step 0 - The raw geocoding API

Whenever I do anything with web services, I always add a request caching layer. So here are the base building blocks implemented in 2 functions. One for doing request caching and the second to do the actual REST query to the geocoding service.

```
Result['precision']; foreach($xml->Result->children() as $key=>$val) { if(strlen($val)) $ret[(string)$key] = (string)$val; } return $ret;?>
```

The above code is the contents of `geo.inc` which you will see included in the following examples.

Easy enough? No real tricks here. We simply send a regular GET request to <http://api.local.yahoo.com/MapsService/V1/geocode> with the location parameter set to an address. You can try it yourself directly from your browser by clicking here:

<http://api.local.yahoo.com/MapsService/V1/geocode?appid=rlerdorf&location=701%20First%20Ave,%2094089>

You can read more about the geocoding service here:
<http://developer.yahoo.net/maps/rest/V1/geocode.html>

Step 1 - Writing your first application

We can just toss a form around this and dump the results to make sure things are working.

GeoCoding API Example

You can see this one in action here:

<http://lerdorf.com/php/ymap/geo1.php>

Note how it is able to fill in missing details for a partial address. eg.

<http://lerdorf.com/php/ymap/geo1.php?location=701+First+Avenue+94089>
results in:

```
[precision] => address  
[Latitude] => 37.416384  
[Longitude] => -122.024853  
[Address] => 701 FIRST AVE  
[City] => SUNNYVALE  
[State] => CA  
[Zip] => 94089-1019
```

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

[Country] => US

This means that you can use it for a bunch of different things. Address to lat/long, of course, but also address to city, or city to zip code conversions. Or 5-digit zip to 5+4. This is of course rather US-centric right now, but that will improve over time.

Step 2 - Adding a map

The geocoding is cool, but an actual map is cooler. Easy enough:

```
#mapContainer { height: 600px; width: 800px; } GeoCoding API Example mymap.addMarkerByLatLon( new CustomP  
OIMarker("  
", "", '0x0012f0', '0xFFFFF'), new LatLon());
```

You can also let the API figure out your markers for you which makes this even simpler. If the RSS feed is using georss correctly you can use the GeoRSSOverlay mechanism. Here it is using the earthquake RSS feed directly:

<http://lerdorf.com/php/ymap/rssquakes.php>

And here is the code. I am still loading the RSS feed myself from PHP because I want to get the pubDate and title from it, but everything else is handled automatically.

```
#mapContainer { height: 600px; width: 800px; }
```

Posted by Rasmus in Software at 15:32