Using PHP classes to navigate distributed whois databases

By Mark Jeftovic

Table of Contents

Introduction

When we think of conducting whois lookups we're probably used to that meaning: querying the Internic database, usually for a domain record for a .com/.net/.org or .edu domain.

We also knew that there were other whois servers out there, one for ..mil and .int for example, not to mention the ones for the myriad ccTLD's (country code TLD's) in existence, and we could look up domains in those TLD's by asking the approprate whois server about it.

Things changed radically this year with the introduction of competition at the registrar level in the .com/.net/.org namespace and with it the advent of the SRS (Shared Registry System). ICANN set up shop to succeed the IANA and began accrediting other registrars while Network Solutions split into seperate business units to handle their registrar operation (which now competes with all the others, albeit with a nice head start) and the registry, which is still for all intents and purposes, a monopoly.

After a period of holding out and refusing to sign the ICANN registrar accreditation agreement, Network Solutions finally did so and as per the terms of the deal, began redirecting port 43 whois service from rs.internic.net over to whois.nsiregistry.net on December 1st/1999.

What all this means is that on December 1st, running "whois devshed.com" stopped looking like this:

InfoWest Global Internet Services, Inc. (DEVSHED−DOM) 1845 W. Sunset Blvd. St. George, UT 84770 US Domain Name: DEVSHED.COM Administrative Contact: Cosby, David R (DRC4) dcosby@INFOWEST.COM 801.674.0165 Technical Contact, Zone Contact: Gifford, Aaron D (AG44) agifford@INFOWEST.COM 435−674−0165 (801) 634−9567 (FAX) 435−674−9654 Billing Contact: InfoWest Domain Services (IDS2−ORG) dns@INFOWEST.COM 435−674−0165 Fax− 435−674−9654 Record last updated on 06−Aug−1997. Record created on 06−Aug−1997. Database last updated on 4−Dec−1999 12:14:06 EST. Domain servers in listed order: NS1.INFOWEST.COM 204.17.177.10 NS2.INFOWEST.COM 204.17.177.20

And started looking like this:

Domain Name: DEVSHED.COM Registrar: NETWORK SOLUTIONS, INC. Whois Server: whois.networksolutions.com Referral URL: www.networksolutions.com Name Server: NS1.INFOWEST.COM Name Server: NS2.INFOWEST.COM

Basically, the default whois server ceased to be whois.internic.net (a.k.a rs.internic.net) which only carried records for domains registered via Network Solutions or Worldnic, and started to be the registry whois server which carried entries for all .com/.net/.org domains.

Overview of the whois landscape

Excepting for the moment .com/.net/.org, most TLD's have a whois server somewhere listening on port 43 where you can connect to it, give it a domain name, and get a record back for that domain name if it exists. The formats of the returned record vary across different servers.

The .com/.net/.org namespace is a little more convoluted. We have the central registry database at whois.nsiregistry.net which contains a minimal record for all domains in this namespace, and then each registrar operates it's own whois server with more complete records for only the domains they administer. Further, each registrar has it's own whois output format for their own records.

The result of all this is One Big Mess. Depending on whether you want to look up a ccTLD or a gTLD (generic TLD), and if the latter, who the registrar is and where their whois server is; trying to create a script to integrate this kind of functionality into your website can be, in a word, nightmarish.

Making sense of it with PHP classes

Enter Whois2.php, a collection of PHP classes designed to take the guesswork out of all this and provide tools to the webmaster to lookup and process domain name data without having to worry about where to get it.

Assumptions:

The class will be used primarily for querying domain names. While other query types exist on some servers, notably contact and host records on the Network Solutions' server, most simply support domain name lookups. If we want to do another query type, we'll see that the class still provides us with methods of doing so.

Web whois databases are not supported. If a given TLD's registry doesn't operate a whois server on port 43, it doesn't belong in this class. .TO and .FM for example, only offer whois lookups through a web interface, and as such, there is no method to lookup .to or .fm domains here.

Objectives:

portable

We want to make the class as portable as possible across as much of the namespace as possible. This means we want to create a class in which we don't have to do anything different in our code to lookup a domain whether it's .com or .cz. Of course due to the differences in output we may have to handle those results differently, but getting the data back from a query should be transparent regardless of what the query is.

modular

We want to be able to break tasks down to smaller, managable sub−tasks. And if part of our landscape changes (i.e. the address of a whois server changes, or the output of one is different) we want first, that the change doesn't affect the rest of our operations and second, that it's as easy as possible to revise our class to cope with it.

object oriented

We are lazy, and we don't want to code anything twice, so in breaking down our code into seperate modules for different tasks, those modules should inherit anything they need to know that other modules also need to know from a common parent. The parent or base class should be able to hand off data to other handlers in a seemless, generic way.

Overview of Whois2.php

This package is a lot different from it's precursor whois.php3, which was a single class that had to be tweaked and hacked everytime something changed. Whois2.php is designed in a way that if one registrar somewhere changes their output format (something NetSol does often enough), the entire class won't break.

When all is said and done, we can use this class to do whois lookups for 121 TLDs, including the gTLD's .com/.net/.org/.edu/.int/.mil, the ccTLD's like .ca, .il, .at etc., Nominet's 2LD domains uk.com, gb.com and gb.net, and the ORSC (Open Root Server Consortium) alternate staging roots (.web, .info, .shop, etc).

At the very least we can get back simple output from the whois server itself in raw, unparsed form. In better cases we have extended methods to further process the raw result into nice key/value pairs, and in the case of domains registered in .com/.net/.org we can seperate parsed results into both those of the registry server response and then the corresponding registrar's server output, provided we have the methods available to parse it.

If the methods do not exist to further process data, the class returns what it has. If you want to create your own extended class to provide methods to parse the raw output of a given TLD, it's very simple to add it to the mix.

Looking at the pieces

main.whois

The base class is class "Whois" defined in main.whois. We've intentionally upper−cased the "W" which means that if can begin working with this in your existing scripts, the new class will not collide with anything that may be using the old version 1 whois.php3.

The $Query variable is a hash that functions more or less as the local environment for the object. It keeps track of things like what our current whois server is ($Query["server"]), what the current TLD is ($Query["tld"], not as obvious as you might think at first glance, considering the nominet uk.com and gb.net, which we support), what the actual query is ($Query["string"]) and last but definately not least, an array of any errors encountered in $Query["errstr"].

We invoke the class in the normal method, and pass our query as well:

```
$whois = new Whois("devshed.com");
```

In PHP we can create constructors for our classes by putting a function in the class with the same name as the class. We rely heavily on those here and in our first constructor we basically get everything ready for a normal domain lookup (we don't have to invoke the class this way but for our purposes we'll just look at domain lookups in normal fashion for now).

```
class Whois {
..
..
..
function Whois ($query="") {
require("servers.whois");
$this−>VERSION=sprintf("Whois2.php v%s:%s",
$this−>CODE_VERSION,
$this−>DATA_VERSION);
if(isSet($query)) {
$this−>Query["string"]=strtolower($query);
$tld=$this−>GetTld($this−>Query["string"]);
if($tld) {
$this−>Query["server"]=$this−>DATA[$tld][0];
if(isSet($this−>DATA[$tld][1])) {
$handler=$this−>DATA[$tld][1];
$this−>Query["file"]=sprintf("%s.whois",$handler);
$this−>Query["handler"]=$handler;
}
$this−>Query["tld"]=$tld;
} else {
```

```
$this->Query["status"]=-1;
$this->Query["errstr"][]=$this->Query["string"].
" domain is not supported";
unset($this->Query["server"]);
}
} else {
$this->Query["server"]=$this->NSI_REGISTRY;
}

}
```

So we set the TLD, set the current whois server, and if we have one, we define an extended handler. The whois.main then takes care of the basic tasks that we need to do regardless of what we're doing it to: connect to the server with Connect(), send the query string and read back the output with Lookup(), and then, if we have an extended handler, pass the result to Process().

servers.whois

One of the first orders of business to to load the "servers.whois" file. This contains the $DATA array which is an array indexed by supported TLD's, the values of which are arrays with their corresponding whois server as their first value, and an optional "extended handler" as value 2. We want to keep all this in a seperate file because this data will take on an update history seperate from the code itself, as new whois servers are reported and (hopfully) energetic PHP coders provide additional extended classes for various TLDs and registrars.

extended handlers

The extended handlers demonstrate PHP extended classes and inheritance. Because output varies wildy from server to server, we provide additional processing of raw output via modular and extensible "handlers", which in themselves, can utilize even further handlers.

Take "devshed.com" as our example again. We know from using the information in the $DATA array that it will be queried at whois.nsiregistry.net, the central registry server. We also see that there is an extended handler "gtld" specified for this TLD. So the Process() function is going to load the code in gtld.whois and then invoke an instance of the gtld class which extends the whois class.

```
if(!defined("__GTLD_HANDLER__")) define("__GTLD_HANDLER__",1);

class gtld extends Whois {

var $HANDLER_VERSION = "1.0";
..
..
function gTLD ($data,$query) {
$this->Query=$query;
$this->SUBVERSION = sprintf("%s-%s", $query["handler"],
$this->HANDLER_VERSION);
```

```
$this->result["regyinfo"]=$this->ParseRegInfo($data["rawdata"]);
if($this->HACKS["nsi_referral_loop"]
$this->result["regyinfo"]["whois"]==
$this->HACKS["wrong_netsol_whois"] ) {
$this->Query["server"]=$this->HACKS["real_netsol_whois"];
} else {
$this->Query["server"]=$this->result["regyinfo"]["whois"];
}
$this->result["rawdata"]=$this->Lookup($this->Query["string"]);

$this->Query["handler"] =
$this->REGISTRARS[$this->result["regyinfo"]["registrar"]];
if(!empty($this->Query["handler"])) {
$this->Query["file"]=sprintf("%s.whois",
$this->Query["handler"]);
$this->result["regrinfo"]=$this->Process($this->result["rawdata"]);
}

}
```

Once again there is a class constructor in class gltd, and we parse the registry output in to key/value pairs. It also turns out that the registrar for devshed.com happens to be Network Solutions, and as it so happens, we have yet another extended class "netsol" so we query that server, get the results, and and hand it off to class "netsol" parse it into key/value pairs as well.

The gtld object uses it's inherited Process() and Lookup() methods to do the follow−up query to the appropriate whois server, and if that's it, it would stop there, returning the raw output. But in this case the gtld class does have a "netsol" handler to further process the data, so it hands off to it for parsing.

We can then add further handlers for ICANN accredited registrars by adding them $REGISTRARS array of the gtld, much in the fashion we can add additional ccTLD handlers to the $DATA array.

Tying it all together

When all is said and done, and you simply want to look up pretty well any domain name in your PHP scripts, you simply need do this:

```
$whois = new Whois("devshed.com");
$result = $whois->Lookup();
```

Depending on what we have under the hood as far as extended handlers go, $result would contain:

In the case of no extended handlers:

```
$result["rawdata"]
```

will contain an array of the raw output from the whois server.

In the case of a .com/.net/.org domain with *NO* registrar handler:

```
$result["regyinfo"]
```

will contain parse key/value pairs of the registry output, including an array of "nameservers" and

```
$result["rawdata"]
```

will contain the unparsed raw output from the whois server defined in $result["regyinfo"]["whois"]

....and in the case of a .com/.net/.org/.edu that has a registrar handler you'll have the two as above plus $result["regrinfo"] which will be an array with key/value pairs from the registrar whois server.

To see exactly what we have in our $result or any parts therein we can use the handy−dandy showObject() function in utils.whois, which is bundled for debugging and diagnostics purposes (along with a couple of other goodies.)

So, to lookup devshed.com and then examine the actual object we get back we can do this:

```
$whois = new Whois("devshed.com");
$result = $whois->Lookup();
```

```
include( "utils.whois");
$display = new utils;
$display->ShowObject($result);
```

And the final result is:
A $result["regyinfo"] array with all the registry info, note that the nameserver list itself another array in $result["regyinfo"]["nameserver"]:

```
regyinfo->Array
domain->DEVSHED.COM
registrar->NETWORK SOLUTIONS, INC.
whois->whois.networksolutions.com
referrer->www.networksolutions.com
nameserver->Array
0->NS1.INFOWEST.COM
1->NS2.INFOWEST.COM
```

A $result["rawdata"] array that has the raw data from the registrar's whois server.

```
rawdata->Array
0->The Data in Network Solutions' WHOIS database is provided by Network
1->Solutions for information purposes, and to assist persons in obtaining
2->information about or related to a domain name registration record.
3->Network Solutions does not guarantee its accuracy. By submitting a
4->WHOIS query, you agree that you will use this Data only for lawful
5->purposes and that, under no circumstances will you use this Data to:
6->(1) allow, enable, or otherwise support the transmission of mass
7->unsolicited, commercial advertising or solicitations via e-mail
8->(spam); or (2) enable high volume, automated, electronic processes
9->that apply to Network Solutions (or its systems). Network Solutions
10->reserves the right to modify these terms at any time. By submitting
11->this query, you agree to abide by this policy.
12->
13->Registrant:
14->InfoWest Global Internet Services, Inc. (DEVSHED-DOM)
15-> 1845 W. Sunset Blvd.
16-> St. George, UT 84770
17-> US
18->
19-> Domain Name: DEVSHED.COM
20->
21-> Administrative Contact:
22-> Cosby, David R (DRC4) dcosby@INFOWEST.COM
```

```
23-> 801.674.0165
24-> Technical Contact, Zone Contact:
25-> Gifford, Aaron D (AG44) agifford@INFOWEST.COM
26-> 435-674-0165 (801) 634-9567 (FAX) 435-674-9654
27-> Billing Contact:
28-> InfoWest Domain Services (IDS2-ORG) dns@INFOWEST.COM
29-> 435-674-0165
30->Fax- 435-674-9654
31->
32-> Record last updated on 06-Aug-1997.
33-> Record created on 06-Aug-1997.
34-> Database last updated on 6-Dec-1999 16:08:49 EST.
35->
36-> Domain servers in listed order:
37->
38-> NS1.INFOWEST.COM 204.17.177.10
39-> NS2.INFOWEST.COM 204.17.177.20
40->
```

And finally a $result["regrinfo"] array, because we have a "netsol" handler to parse output from the Network Solutions registrar server we have the following data:

```
regrinfo->Array
organization->InfoWest Global Internet Services, Inc.
org_handle->DEVSHED-DOM
org_address->1845 W. Sunset Blvd.
St. George, UT 84770
US
domain->DEVSHED.COM
admin->Array
name-> Cosby, David R
handle->DRC4
email->dcosby@INFOWEST.COM
tech->Array
name-> Gifford, Aaron D
handle->AG44
email->agifford@INFOWEST.COM
billing->Array
name-> InfoWest Domain Services
handle->IDS2-ORG
email->dns@INFOWEST.COM
updated->06-Aug-1997
created->06-Aug-1997
db_updated->6-Dec-1999 16:08:49 EST
ns->Array
NS1.INFOWEST.COM->204.17.177.10
NS2.INFOWEST.COM->204.17.177.20
```

At the time of writing the only extended handler for registrar's is the netsol class. We need further classes for register.com, melbourneIT, COREnic and all the other registrars. Until then, our output would have stopped with the $result["rawdata"] array above.

Another Example:

Looking up something other than a domain name:

```
$STRING = "MJ177";
require("whois2.php3");
$whois = new Whois();
$whois->Query["server"]="whois.networksolutions.com";
$result = $whois->Lookup($STRING);
```

$STRING in this case is a Network Solutions contact handle (mine). It could just as easily be a host handle or any other string. The registrar whois servers at NetSol allow wildcard matching on arbitrary strings.

Conclusion

Hopefully this article has served a two−fold purpose, first to explain the new layout of the whois servers in light of newly accredited registrars and the SRS. Second, to demonstrate the rudiments of using PHP classes, which lend themselves quite nicely to the diverse and ever−changing situtation we have applied them to here.

The latest vesion of Whois2.php can be downloaded from http://www.easydns.com/~markjr/whois2/