PHP and SOAP - First Steps

(lou wilkinson, louwilkinson1@gmail.com)

--Intro--

This is the beginner's, beginner intro for getting SOAP working with PHP. Quite a bit more basic than most, simply because it's all new to me and I wanted to write it all down and, possibly, save you the same headache's I've had the last couple days.

Also, it's not a read along...it's a do along. So roll up your sleeves and good luck.

If you're like me, then all you really need to do to get started is see one simple example, explaining the parts and how they fit. Once you see how something works, start to finish, you're ready to go...so that's what this is....one simple example, explained, as best I can remember, start to finish.

Whether is Xampp, Lampp, Zend, iAmp, or any other php installation, the first script you should write is showphp.php

```
<?php
        phpinfo();
?>
```

Run it, look it over, if your installation is working, this'll show you what you've got. If it doesn't run and show you all the details of your php installation, then you've got problems more basic that what we're going to deal with here....so back up a bit.

The part we're interested in, today, is the SOAP section near the bottom. You want to be sure that it's compiled in / turned on.

soap

Soap Client	enabled

Now, given that you have a working php installation and, presumably, some type of editor for writing your php scripts, you're going to need two more things to really get started.

1) a SOAP server that will respond to requests.
2) SoapUI, a great program to let you actually see the requests and responses.

--Installation and Setup--

1) xmethods.

Some nice folks that have put up a page of SOAP services that can be freely used to do EXACTLY what we're trying to do here....get a working model going.

http://www.xmethods.net/ve2/index.po

and for this walkthrough, we're going to use the StockQuote service because it IS so simple.

http://www.xmethods.net/ve2/ViewListing.po;jsessionid=QpWEL-0yf9Fd6gkBMju2DOZb?key=467541

So go to the site, scroll down until you find the StockQuote service and click the link.

You should see, somewhere on the page a url that ends with ?wsdl **that's what we want**, that whole url, including the ?wsdl part.

http://www.restfulwebservices.net/wcf/StockQuoteService.svc?wsdl

For right now, just hang on to it. We're going to need it a couple times farther along in our experiments.

Before going on to the next step, you might want to click the *Analyze WSDL* link on the page....this should show you what kinds of things you can request from this service.

Operation / Method Name	SOAPAction
GetStockQuote	GetStockQuote
GetWorldMajorIndices	GetWorldMajorIndices

You might even, if you were the curious sort, go trolling around on the xmethods site, looking for other services you can play with.

Now, before we leave xmethods, click on the actual wsdl link itself...the one that ends with ?wsdl. Wow. What a mess. And remember that this is a simple example!

So that's what we're trying to avoid....we're going to use some really nifty things in php that let us kind of blow over 99% of that and just get what we want.

Anyway, moving on....

2) soapUI.

More nice folks. These folks have put up a software package that you can download for free (or pay for the professional version) and actually SEE the requests that you send and receive.

So here, in a nutshell, is what we're going to do: we're going to use soapUI to get a working service request going with xmethods. Then we're going to examine the service request and make our php look just the same....then run our php and get our data back. Simple.

So download and install the soapUI package. I'll meet you back here when you're finished.

(final jeopardy theme music here....)

Ok...great.

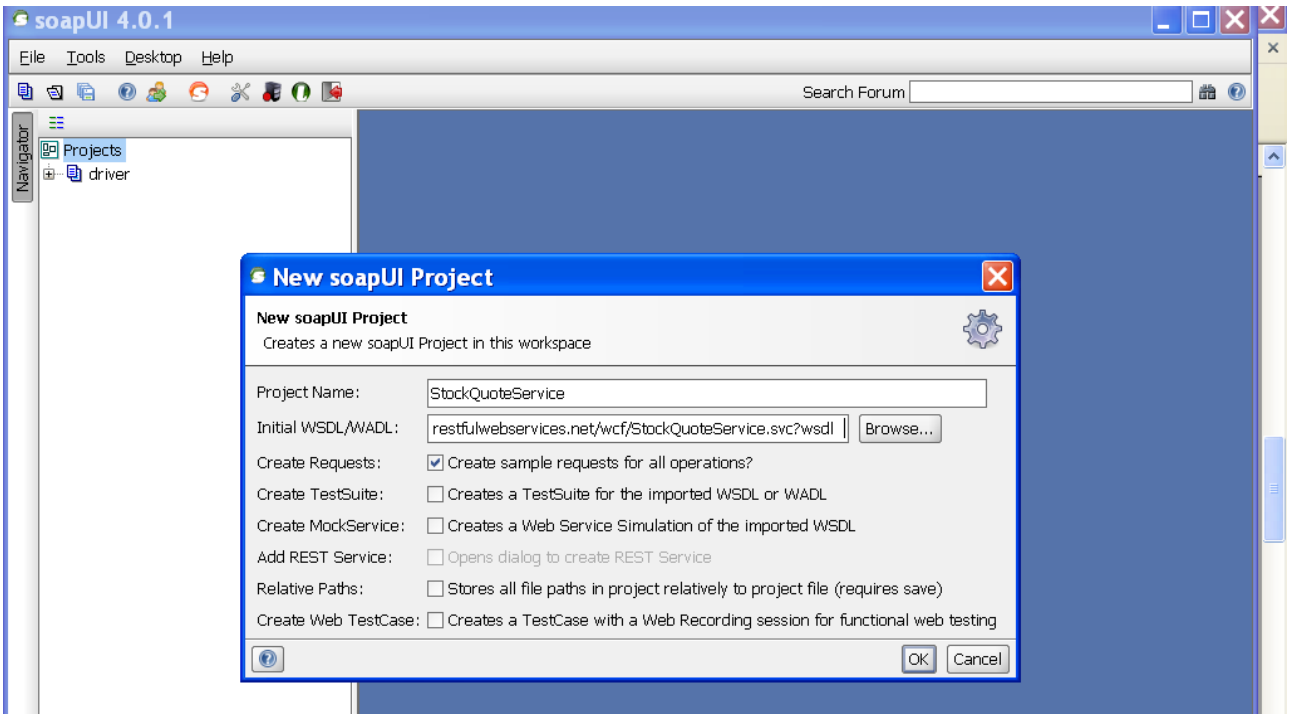
First, go find that link you saved from before...the one that ends with ?wsdl.

Open up soapUI, hit File / New soapUI project.

A little window should open and, in the 2nd slot, the one that says “Initial WSDL/WADL” put in the link you've been saving.

IMPORTANT: make sure there are no extra spaces at the beginning or end of the link.

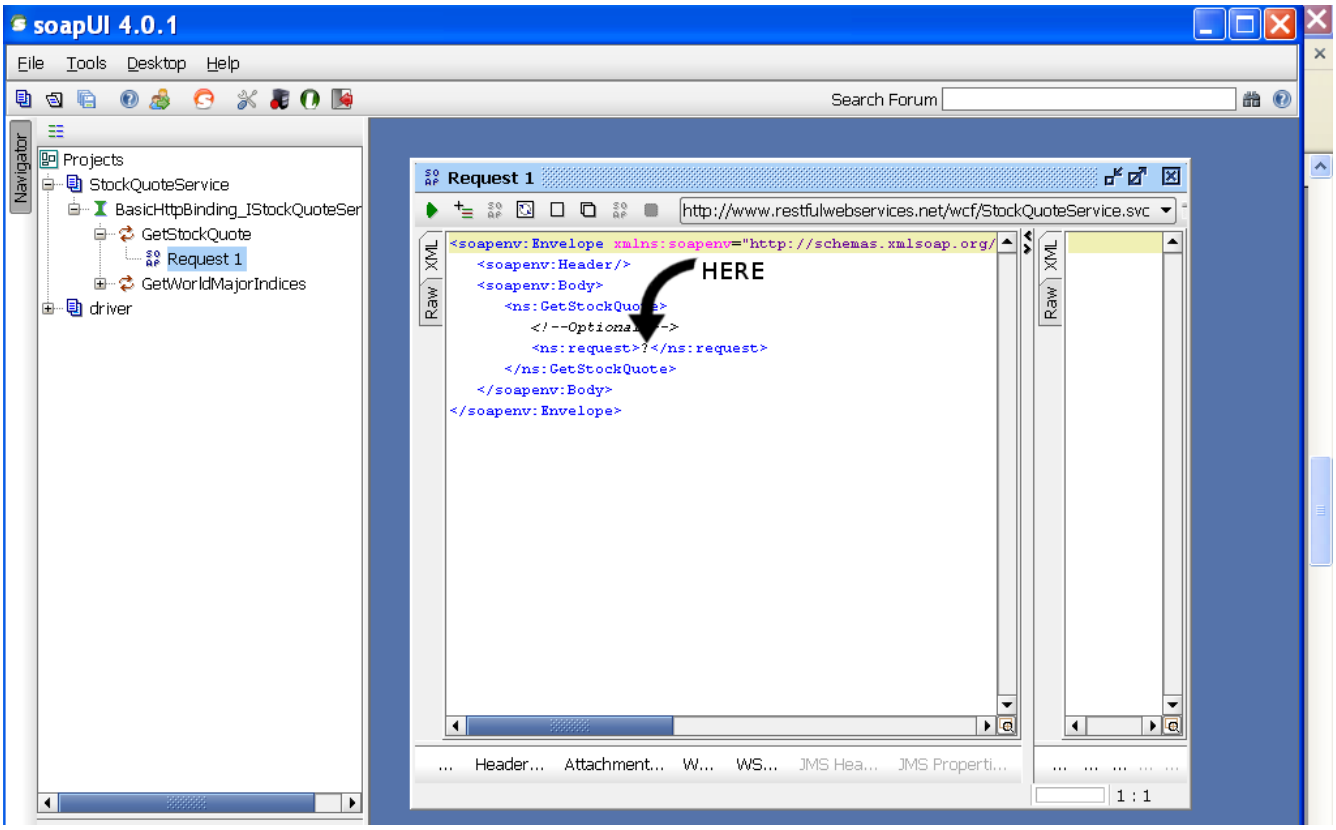
SoapUI should autofill the title, put check marks in boxes, etc until it looks like this:



Now hit the OK button...soapUI goes to that site, reads the wsdl document and figures everything out...it'll take a couple seconds and then you'll see that it's filled in a project in the left hand pane.

Open StockQuoteService / BasicHTTP.../GetStockQuote and you should see Request 1...double click it to open.

This is the actual XML file that you're going to "send". Look it over....the structure should kind of make sense...header stuff, body stuff, then you see the actual <GetStockQuote> request...and inside the request a couple of tags with a question mark between them.



Blank **out** the **?**, type in **IBM** so that line looks like this:

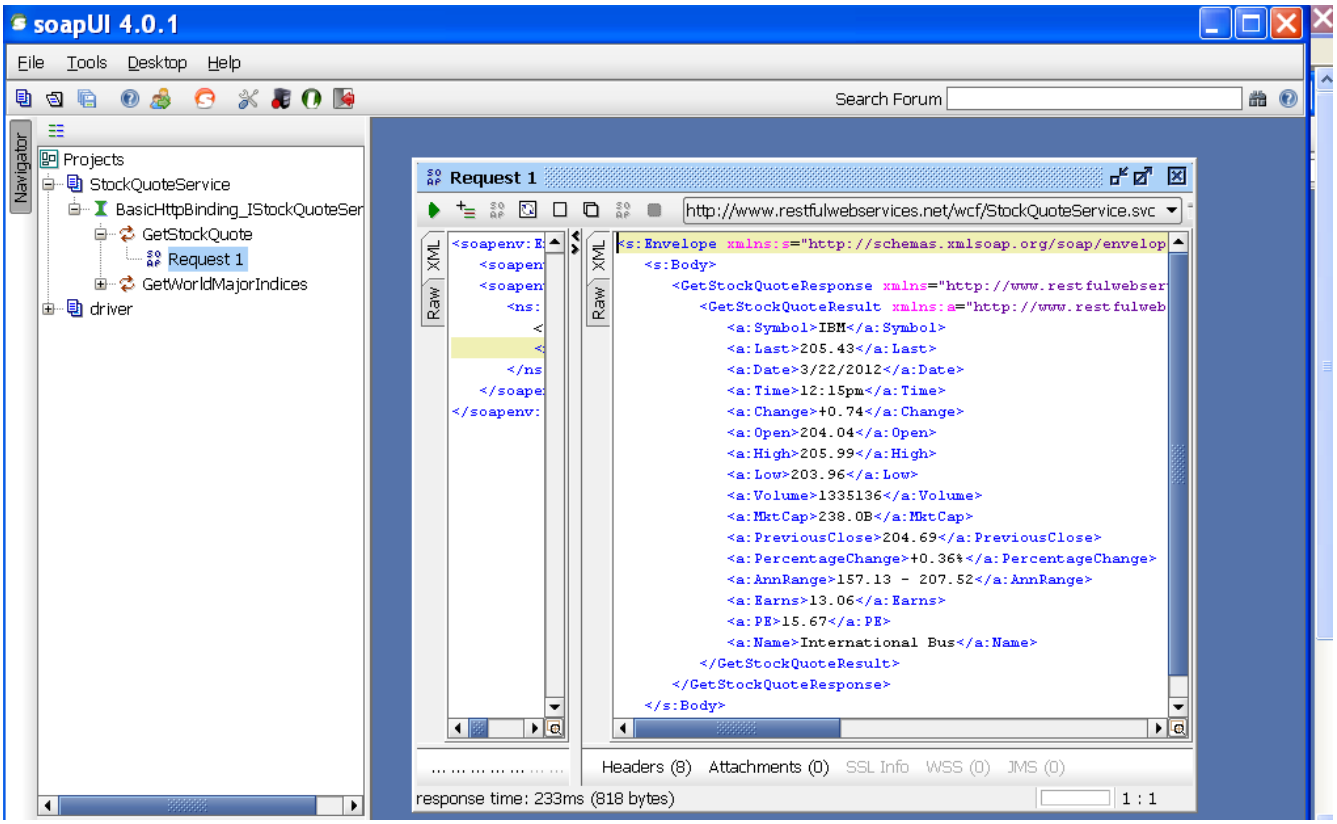
<ns:request>IBM</ns:request>

Take another look...this is the exact request you're going to send to the remote soap server.

(I don't want to get ahead of ourselves...but notice that the namespace (ns) for the **IBM** text that you're sending is request. We're going to come back to that in a minute.

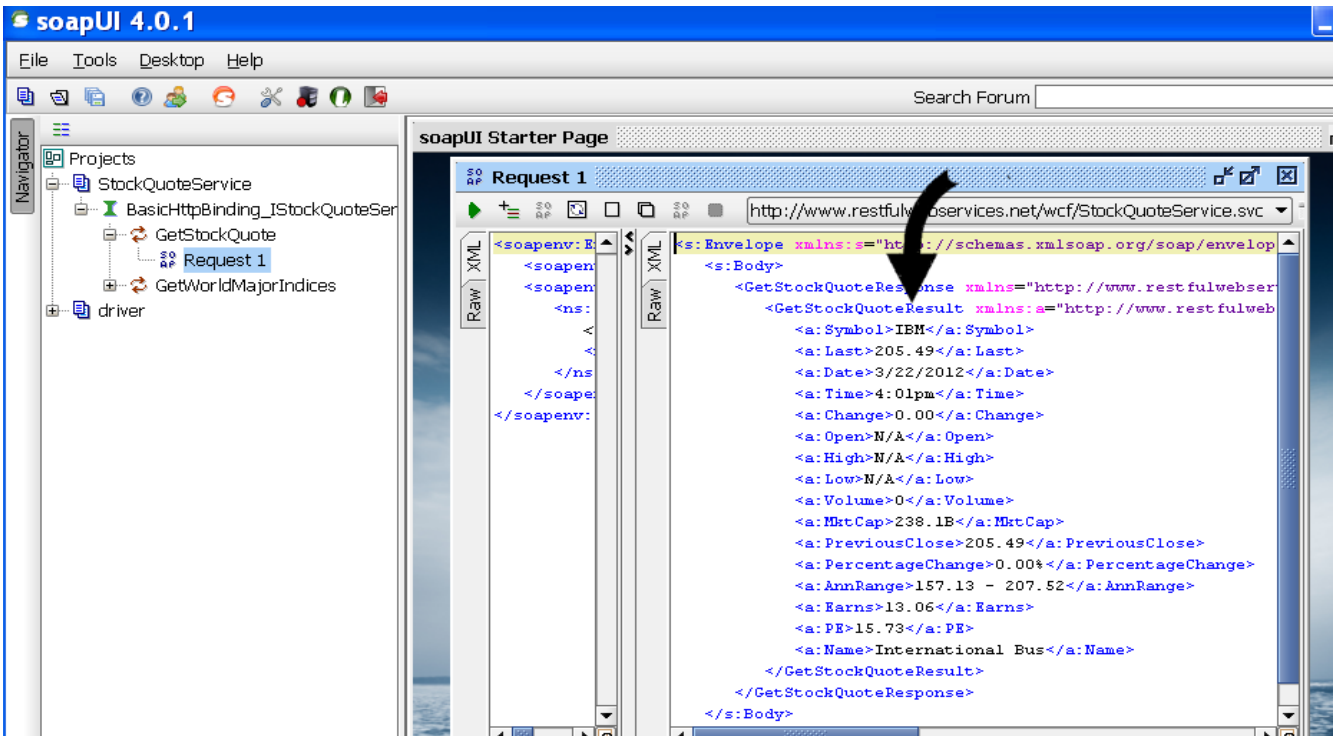
Click the little green triangle/arrow in the upper left corner of the panel to send your request.

Patience....patience... There it is. Your first soap request. You know, Liberace used to have his fingers insured...maybe your boss should be doing the same thing for you!



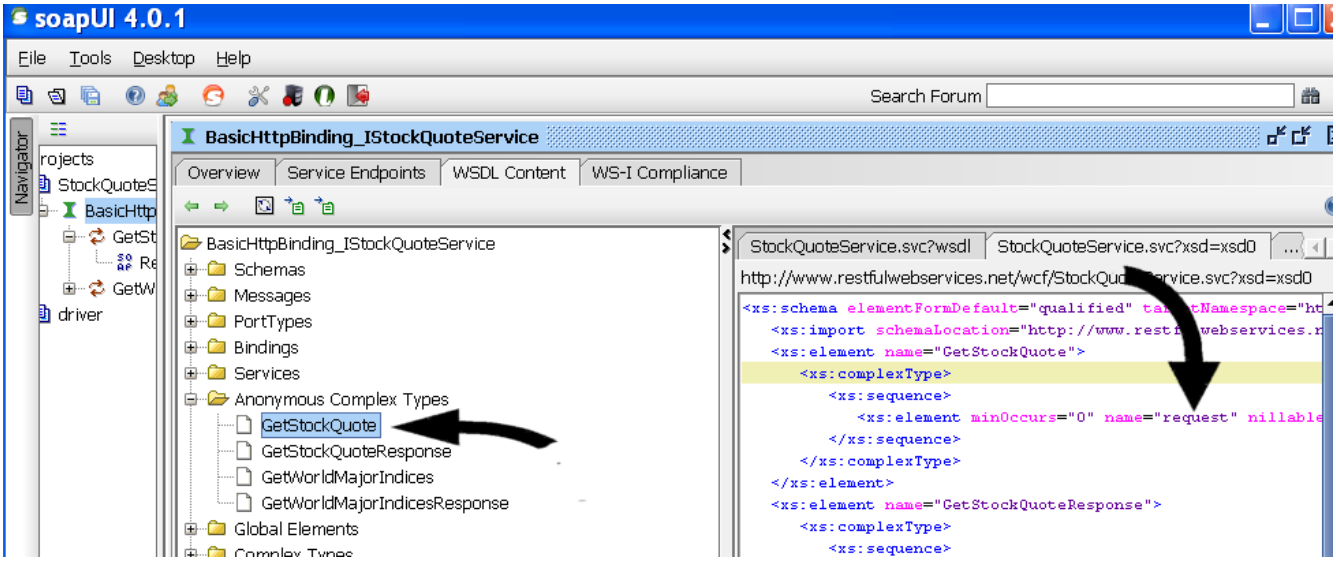
This is the actual XML that the soapserver is going to return to your program.

One more important item. Notice the name of the tags around the returned stuff.



See how it's the same tag as what you sent but with Result added on? That's going to be important. I believe that's the most common return tag, certainly the most common I've seen, but not the only possibility so you always want to check.

Now, last, before we end this part, part 1, click, in the left pane on BasicHttpBinding...and then, when the page comes up, click on the WSDL Content tab. Choose, on the left side, Anonymous Content Types or Global Elements and choose the function your going to invoke (GetStockQuote). Notice the “name” of the element...you're going to need that.



So this ends part 1. All the confusing “what's going on here” stuff. Next comes the easy stuff.

To summarize: you've found a website that will service soap requests...you've used soapUI to, interactively, make a request and gotten a response.

The next part is much shorter...we're going straight to the php program to do, basically, what you just did here with a php script.

You'll need the url for the WDSL we've been using, you need to remember the name of the element you're sending, and you'll need the name of the return tags.

Hopefully, I made enough of a point of it that you know those are:

- http://www.restfulwebservices.net/wcf/StockQuoteService.svc?wsdl
- request
- GetStockQuoteResult

--PHP--

Finally!

Ok, we're going to walk through this line by line...then run the script with debug / visibility stuff added, then strip that out and just let it roll. So let's get started.

```
<?php

$wsdl = "http://www.restfulwebservices.net/wcf/StockQuoteService.svc?wsdl";

$client = new SoapClient($wsdl, array(
                                    "trace"=>1,
                                    "exceptions"=>0));
```

We set a variable to the ?wsdl link we've been using and instantiate a SoapClient object. We also set the options array so that we can get some visibility later in the script if we need it. We'll strip that back out for the final version, but now, while we're testing, i'll come in handy.

```
$stock = "IBM";

$parameters= array("request"=>$stock);

$value = $client->GetStockQuote($parameters);
```

We set our stock symbol to some value, make an associative array and notice that the index is `"request"`....that's important...that's the deal from up above. It has to match the tags of the element in the soap request.

The next line, `$value = $client->`... is really cool. What happens under the hood here is amazing. You're sending a request to the soap server, getting all that xml back, sorting through the xml to find the legitimate methods that can invoked, seeing that one of them is `GetStockQuote`, taking “IBM”, slamming it into the <ns:result> tags, and getting the data back from the soap server.

All in this one instruction. Pretty slick, eh?

Next we're going to see what we've got.

```
print "<pre>\n";

print "<br />\n Request : ".htmlspecialchars($client->__getLastRequest());

print "<br />\n Response: ".htmlspecialchars($client->__getLastResponse());

print "</pre>";

?>
```

Run it. You should see that your request looks an awful lot like what soapUI sent and that your response looks a lot like what soapUI received, albeit all in one string.

If there are errors, this is where you'll see them...instead of a lot of xml with “IBM” stuff embedded, you'll see error messages.

So here's the script

```
<?php

	$wsdl = "http://www.restfulwebservices.net/wcf/StockQuoteService.svc?wsdl";

	$client = new SoapClient($wsdl, array(
					"trace"=>1,
					"exceptions"=>0));
	$stock = "IBM";

	$parameters= array("request"=>$stock);

	$value = $client->GetStockQuote($parameters);

	print "<pre>\n";

	print "<br />\n Request : ".htmlspecialchars($client->__getLastRequest());

	print "<br />\n Response: ".htmlspecialchars($client->__getLastResponse());

	print "</pre>";

?>
```

Now, let's tweak the output, just so we can get a little more data and, just to prove that we can, let's pick NCR instead of IBM.

First, the normal stuff up front...and now that my script is working, notice that I've taken the options array out when instantiating the object.

```
<?php

	$wsdl = "http://www.restfulwebservices.net/wcf/StockQuoteService.svc?wsdl";

	$client = new SoapClient($wsdl);

	$stock = "NCR";

	$parameters= array("request"=>$stock);

	$values = $client->GetStockQuote($parameters);
```

Now let's change what we do with the output after all the magic happens.

```
	$xml = $values->GetStockQuoteResult;

	print "<pre>\n";
	print_r($xml);
	print "</pre>";

	?>
```

Run the script. You should see, in your browser, something like:

```
stdClass Object
(
    [Symbol] => NCR
    [Last] => 20.82
    [Date] => 3/22/2012
    [Time] => 4:05pm
    [Change] => 0.00
    [Open] => N/A
    [High] => N/A
    [Low] => N/A
    [Volume] => 0
    [MktCap] => 3.296B
    [PreviousClose] => 20.82
    [PercentageChange] => 0.00%
    [AnnRange] => 15.28 - 22.12
    [Earns] => 0.329
    [PE] => 63.28
    [Name] => NCR Corporation C
)
```

There are two important things here:

1) you use `GetStockQuoteResult` to actually return the results into something, and
2) that "something", even though it looks like an associative array, is actually an object, `stdClass Object,` which you'll need to know for the next and final step.

An object is not ALWAYS returned. You need to try something like this experiment to determine what it is that's actually being returned, an object, a simple value, etc.

So keep that in mind....you have to examine what kind of thing is coming back to your program so you can deal with it properly once it's returned. That's why I'm a fan of the print_r() and <pre> bits.

```
print "<pre>\n";
print_r($xml);
print "</pre>";
```

Now, finally, there's only one more little piece that you need before you start having your way, all around town, with various wsdl services.

How to get one piece of that data, when it IS an object, into a normal php variable so you can put it on the screen or in a file or whatever it is that you want to do with it.

And to do that, we just need to look over the returned "stuff".

I'm going to try to snag the current price...and looking over the returned data, I see that it's called "Last", so:

```
$currentprice = $xml->Last;
```

That's pretty much all there is to it.

So let's finish up with a minimal script.

```php
<?php
	$wsdl = "http://www.restfulwebservices.net/wcf/StockQuoteService.svc?wsdl";

	$client = new SoapClient($wsdl);

	$stock = "NCR";

	$parameters= array("request"=>$stock);

	$values = $client->GetStockQuote($parameters);

	$xml = $values->GetStockQuoteResult;

	$currentprice = $xml->Last;

	print "<br />\n Last Value: $currentprice";

?>
```

As you can see, we've got our current stock price comfortably stored in a simple php variable.

Now you've done it. Examined a web service, figured out the methods to send and retrieve the results, unpacked those results into a simple php variable. It all just details from here. Wrapper it with your own throws and catches or whatever you do and you're set to go.

--Credit Where Credit is Due--

A lot of folks have a finger in this pie. There's the previously named folks at xmethods and soapUI that have created these great resources for us to explore wsdl and soap.

There a genius or two somewhere creating php stuff that, somehow, managed to figure out how to unwind all that wsdl into an object that you can simply reference and use...I can't imagine how much aspirin THEY went through!

There are the folks at Zend, who's forum I trolled and read and trolled and read and trolled...

And finally, there's someone named A.Mitchell who's document, finally, put it all together for me. **http://metrix.fcny.org/wiki/display/tips/How+to+Create+a+PHP+Client+for+a+.NET+and+SOAP-based+Web+Service+API**