Interview – PHP's Creator, Rasmus Lerdorf

The membership of the SitePoint community forums recently got together and produced a bunch of questions for PHP's original creator, Rasmus Lerdorf. In reviewing his responses, I was pleased to discover that the man who originally put the PHP machine in motion maintains an unclouded vision of what the open source movement is all about.

He is quick to play down his contribution to what PHP is today, instead attributing most of PHP's success to the vast community of developers that have signed on to the project over the years. In a sense, Rasmus today is simply PHP's biggest fan.

But enough from me; let's hear what Rasmus had to say!

In the beginning…

SP: What was your first contact with the Open Source movement, and what is it about Open Source that got you hooked?

RL: Well, back in the early and mid-90's the term "Open Source" did not exist.

"Free Software" existed, of course, and I had been playing with Linux almost since the very first release in 1991. Previously I was using QNX and Xenix and then started to fiddle with Minix until Linux rescued me.

I don't think I was ever really "hooked" by a "movement". When you don't have the money to buy SCO Unix and you can download something that works and even find people who can help you get it up and running, how can you beat that? Religion never really played a part.

SP: What led you to develop PHP? And what do you think this language has to offer that others don't?

RL: The first version of PHP was a simple set of tools that I put together for my Website and for a couple of projects. One tool did some fancy hit logging to an mSQL database, another acted as a form data interpreter. I ended up with about 30 different little CGI programs written in C before I got sick of it, and combined all of them into a single C library. I then wrote a very simple parser that would pick tags out of HTML files and replace them with the output of the corresponding functions in the C library.

The simple parser slowly grew to include conditional tags, then loop tags, functions, etc. At no point did I think I was writing a scripting language. I was simply adding a little bit of functionality to the macro replacement parser. I was still writing all my real business logic in C.

In the end, what I think set PHP apart in the early days, and still does today, is that it always tries to find the shortest path to solving the Web problem. It does not try to be a general-purpose scripting language and anybody who's looking to solve a Web problem will usually find a very direct solution through PHP. Many of the alternatives that claim to solve the Web problem are just too complex. When you need something up and working by Friday so you don't have to spend all weekend leafing through 800-page manuals, PHP starts to look pretty good.

SP: Looking at the usage figures, there are now over 9 million domains using PHP. Did you have any idea that PHP was going to become this big? How does it feel to know that your product is probably the best alternative to Microsoft's solutions for the Web?

RL: First, to be clear, I did not develop the PHP we know today. Dozens, if not hundreds of people, developed PHP. I was simply the first developer.

PHP is very much a collaborative project. Think of it this way: you have a Web problem. You can either go to the store and buy an expensive shrink-wrapped product that may or may not solve most of your problem. Or you can get together with a couple of thousand people who have the exact same problem as you, and work out a solution that works for all of you.

Not only will you get a solution that addresses your exact problem, you'll also become part of a like-minded community where ideas and experiences flow freely. That beats any commercial product you can go buy at a store, and to me is the best way to develop this type of software.

So when people ask me what it feels like to have developed something that millions of people use, it doesn't really fit with how I view things. In the end, I am simply the first member of a community that has arisen around one approach to solving the Web problem.

SP: Who would you call your hero? Which people in or outside of IT have inspired you?

RL: I don't really get inspired by people in the metaphysical sense. But I definitely appreciate and respect a slick solution to a tough problem.

SP: During your years of PHP development, what do you think was the most important decision you had to make? Are there any decisions you made that you now wish you had decided differently?

It is tough to ask me to second-guess decisions that were made 6 or 7 years ago when PHP was used by a grand total of 1 person. Don't forget that I did not sit down to write a scripting language that would be used by 9 million domains: I sat down to solve a problem. Solving the problem by 5pm so you can go to a movie with your girlfriend leads to some aspects that aren't ideal 7 years later, when thousands of people have to work around that late-night hack you added.

The most important decision I made along the way was probably to give up control. To open up the project and give just about anybody who asked full access to the PHP sources. This brought in a lot of excellent talent, and people tended to feel a real sense of ownership. The PHP project

is probably one of the biggest out there when it comes to the number of people with commit access to the CVS repository where the code and documentation lives.

The Open Source movement

SP: The Open Source movement is still portrayed by many as "anarchistic" and some kind of "threat to society". Do you feel it will ever gain mainstream acceptance? If it does, how will the Open Source community deal with that?

RL: I guess there are two parts to this question.

As far as being mainstream, the product of this "movement" is most definitely mainstream. The "movement" built the Internet as we know it today. It built the TCP/IP stacks used in most of the operating systems people use (yes, even Windows). It built the most popular Web server in the world, along with the DNS and MTA systems that make the Internet tick. Heck, if you go back a bit, it built the entire industry. The first operating systems were all open source, because that was the only sane way to do things. You could not sell someone a big mainframe without providing the source to the brains of the thing. It was only later on that the concept of not providing the source code was introduced.

But I guess your real question is what I think of Microsoft's attempt to convince the world that large groups of people collaborating to solve problems somehow threatens the very fabric of the society we live in. And I don't think there are "many" people making this claim, as it is complete crap — I would like to think that the world is a good place and not full of people who would propagate such a ridiculous idea. Let's put an end to all meetings of large groups of people while we are at it. They might be evil anarchists out to destroy the world.

In the end, mainstream acceptance is not the goal. The goal for most people who work on free software and open source projects is the technology itself. It is building a tool that solves the problem. It is not about ideology for most of us, and as such, mainstream acceptance only involves mainstream use of the technology. This has been achieved on many fronts already, with many more still to come.

SP: PHP gets very little mention in the mainstream IT press. Do you feel that PHP is being deliberately ignored outside of Open Source circles?

RL: PHP is not very exciting and there isn't actually much to it. It is a thin glue layer between your Web server and all the various things you might want your web server to talk to.

In the old tradition of UNIX, we rely on small specialized add-on libraries to do all the heavy lifting with as little interference from PHP as possible. ASP, JSP and Cold Fusion all have large companies with large advertising budgets behind them and the products themselves get bigger and more complex with every release so that customers will feel they got their money's worth. Who is going to spend $10,000 for a floppy and a 2-page manual?

That floppy and the 2-page manual might actually be exactly what they need to solve their problem, and as such it might very well be worth spending $10,000 on a small targeted solution like that. Small targeted solutions are of little interest to large software companies. The concept doesn't scale. Small targeted solutions with no advertising budget are of little interest to the trade rags.

So no, I don't think it is deliberate that PHP gets very little press. PHP is about as exciting as your toothbrush. You use it every day, it does the job, it is a simple tool, so what? Who would want to read about toothbrushes?

SP: The move away from GNU public license from PHP v3.0 to v4.0 has caused a stir among the Open Source community. Do you feel the new licensing model has now been accepted and understood as the best direction for PHP to take?

RL: PHP 3 was dual licensed actually. So we didn't actually move from the GPL to something else, we simply dropped the GPL part.

I don't really see the point of dual-licensing, and it causes a lot of confusion. By putting PHP solely under the Apache-style license it is under today, a lot of this confusion was resolved.

Dual-licensing doesn't really work as far as I am concerned — the less restrictive of the 2 licenses is what people are going to use anyway. The various protections that the GPL offers are quite pointless when people can simply choose to use the software under the less restrictive Apache-style license. So it made sense to just go with the less restrictive of the two licenses.

If you take a look around, there are no significant scripting languages that are GPLed. And by GPLed I mean strictly GPLed in the single license sense. Perl is dual-licensed with the completely unrestrictive artistic license. Python has its own license. Ruby is dual-licensed with its own license. Tcl is under a BSD-style license. I don't see why PHP not being under the GPL would upset anybody.

Like the other scripting languages probably realized somewhere along the line, the GPL is not really needed. I would have no problems with Microsoft abandoning ASP and moving completely to PHP. They might embrace and extend it, sure, but at that point we would be in a purely technical race with them. That's a fight we can win, and in the end, having PHP everywhere would be a cool thing for the PHP community.

PHP today

SP: To what would you attribute PHP's success? Do you feel that PHP has any major weaknesses (in comparison to other languages)?

RL: People like PHP because it solves their Web problem. As such, I don't see any weaknesses. It does the job it was designed to do.

Some people might argue that certain aspects of PHP are not as mature as those in other languages. The OOP support in PHP is an example. But in the end this has very little to do with solving the Web problem and more to do with aesthetics and language purism.

SP: Are you still actively involved in PHP development today?

I am still quite involved. I don't spend 20 hours a day on it like I did in the first couple of years, but I still fix bugs, argue with the other developers about features, and occasionally jump in and add the odd new bit here and there.

SP: Which Web server runs PHP the best? Apache – or something else? And which platform runs PHP the best? Linux/Intel, Solaris/SPARC, or another?

RL: This all comes down to what gets the most attention, I think. Most people use Linux/Intel with Apache. This means that bugs on that platform are discovered by the developers themselves early on, and the end-user is unlikely to hit something that the developers haven't run into already. Other mainstream UNIX platforms such as Solaris/SPARC and FreeBSD/Intel with Apache are right up there as well.

SP: PHP is usually paired with MySQL. How much co-operation is there between the two teams in terms of development?

RL: We know the MySQL folks very well. The first database code in PHP was written for the MySQL predecessor called mSQL. The MySQL API was completely compatible with mSQL's when it came out, so right from the early days of MySQL, PHP had good support for it. The pairing works because PHP and MySQL tend to take a minimalistic and very direct approach to solving problems.

In terms of cooperation at the development level, there isn't that much actually. But not much is needed. PHP provides a thin layer that simply exposes the MySQL API to the PHP user. We bundle the MySQL client library with PHP, but that library is completely maintained by the MySQL team with little involvement from us — except when they break the build, of course.

SP: Do you think PHP is becoming a replacement for Perl?

No, Perl is a general-purpose scripting language. PHP is specifically geared to the Web problem.

SP: What are your views on Magic Quotes and Register Globals?

RL: Register Globals is one of the features that brought people to PHP. The simplicity of creating Web applications when form and other variables were automatically available could not be beaten.

I was personally not in favour of turning Register Globals off by default. It adds very little to the overall security of an application. If people do not check data coming from the user then with or without Register Globals enabled that application is going to be insecure.

The only time having Register Globals off helps is when you forget to initialize a variable before you use it and someone who knows your code exploits that. By changing the error reporting level you can have PHP find these cases for you automatically. So in the end, all I think turning Register Globals off has done is make writing PHP apps more complicated.

And it has of course also generated 10-20 questions/bug reports per day from users who are confused about this change.

Magic Quotes stems from the days when PHP was used almost exclusively for database-driven applications. These applications would take form input and stick it into a database. Even today, a large chunk of the PHP scripts out there do little more than this.

You always have to escape quotes before you can insert a string into a database. If you don't, you get an ugly SQL error and your application doesn't work. After explaining this simple fact to people for the 50th time one day I finally got fed up and had PHP do the escaping on the fly. This way the applications would work and the worst that would happen is that someone would see an extra on the screen when they output the data directly instead of sticking it into the database.

Often people didn't even notice this extra since it did not cause any fatal SQL errors and thus I wouldn't get confused emails asking me what was going on. This was a very good thing.

Even today you still see the odd site where it is obvious that the author didn't realize that data needed to be escaped before being inserted into a DB, and you see the odd extra here and there. Each of those is a support message we didn't have to answer.

The clueful who don't like this feature can simply turn it off and handle all escaping themselves. And the clueful who write portable apps can simply check the setting using get_magic_quotes_gpc() and add an addslashes() call when appropriate.

SP: Do you think there is a successful balance between the commercial and open source elements of the PHP community?

RL: I think it works out ok. The various commercial entities pay individuals to work on parts of PHP — and that benefits everybody.

SP: What's been the most surprising or innovative use of PHP you've seen on the Internet?

RL: I keep seeing new and weird things, the latest being Wez Furlong's ActiveScript SAPI module, which lets you do client-side PHP like this:

```
<html>

 ...

 <script language="ActivePHP">
```

```
  function clickit() {

    $GLOBALS["window"]->open("http://www.php.net");

  }

</script>

...

<img src="..." onclick="clickit();" />

</html>
```

Alan Knowles' PHPMole IDE for PHP written in PHP-GTK is quite impressive as well. There are plenty of other cool PHP things out there, but these are probably the furthest from what I started out doing.

The Future of PHP

SP: Are there any plans for server-side, stateful variables in PHP? It would be useful to place instantiated objects in shared memory so that users didn't have to incur large overhead due to class instantiation.

RL: The Alternative PHP Cache (APC) project can already stick instantiated classes in shared memory, so that one has been solved by APC and others.

Personally I think if you really have performance issues, you should either simplify your code a bit or have a look at writing the critical parts in C. Extending PHP at the C level is actually much easier than most people think.

SP: Current 'session' variables use disk space (e.g. /tmp) which is no good for high-traffic sites. Are there plans to remedy this?

RL: Right from day one of the session support in PHP, we provided a shared memory backend session handler. Just set your handler to mm instead of files in php.ini. However, for high-traffic sites this is not the solution. The real solution is to load-balance the site across multiple servers.

Having session data in memory on a single machine doesn't solve anything. For this, you write yourself a session save handler and stick your session data into a central database of some sort. See http://php.net/session_set_save_handler.

SP: What about database connection pooling? Persistent connections are not nearly good enough – are there plans to implement connection pooling in the future?

RL: A pool of connections has to be owned by a single process. Since most people use the Apache Web server, which is a multi-process pre-forking server, there is simply no way that PHP can do this connection pooling. It has to be done by a dedicated standalone process and is quite outside the scope of PHP itself. Both SQLRelay and SRM can be used to solve this problem.

If/when the common architecture for PHP is a single-process multithreaded Web server, we might consider putting this functionality into PHP itself, but until that day it really doesn't make much sense. Even Apache 2 is still going to be a multi-process server with each process being able to carry multiple threads. So we could potentially have a pool of connections in each process.

SP: Are there plans to improve OOP? Users feel there should be less overhead in instantiating classes; and the provision of encapsulation so that they can keep member variables hidden (promotes better programming). Are there any plans to this effect in the pipeline?

RL: Yes.

SP: Sybase and MS SQL Server provide support for multiple result sets returned from SQL stored procedures. PHP does not support this! When can users expect it?

RL: When someone contributes the code to do it.

SP: Database queries are currently buffered in memory before being available to the client. Can PHP programmers expect this behaviour to change so that queries are available immediately as rows are being sent from the server, so they do not have to wait?

RL: When the various database API's support this, sure they can. We already support this with MySQL through the mysql_unbuffered_query() call and have for quite a while.

SP: ZendEngine2 has plans for a number of exciting new features, such as Exception handling and mature OOP support. Can you give us a rough estimate as to when PHP users can expect a release – at least in terms of months or years?

RL: Nope. It will be released when it is ready.

SP: Do you see a future in PHP-GTK, with popular desktop applications being written in PHP?

RL: I see PHP-GTK mainly being used in cases where you need to provide both a Web interface and a GUI interface to the same application. Being able to use the same backend code for both is a big win.

SP: Is development of PHP and Apache now running in parallel? And is it likely that the two projects will merge in some way, in the future?

RL: PHP and Apache have always been quite closely linked since they are both pieces of the puzzle that solves the Web problem. As such there are a number of developers who work on both projects. But no, the projects will definitely not merge. That wouldn't make much sense.

SP: Do you think major corporations will use PHP in their environments instead of J2EE and .NET in the future?

RL: Some do, so yes.

SP: What would you say to young developers who are considering starting an Open Source project themselves?

RL: I am not sure it is possible in that sense. Sort of like sitting around watching your phone, trying to will it to ring. It always rings when you are in the shower or at some other inconvenient time.

I don't think you really sit down and decide to start an open source project. This "movement" is a bit of a myth that people like to glamorize and assign all sorts of unrealistic characteristics to. Nobody is going to join your project if all you have is a cool idea. Everyone has cool ideas.

People will pool around your efforts if you build something that is useful enough that they find it easier to take your code and extend it a little bit to solve their problem. If your stuff is half-baked then people are likely to dismiss what you have done, and just solve their problem themselves or by using something else out there.

So to start an open source project, first assume that you are completely on your own and solve some problem that has been bugging you for a while. That means months and months of work to get something that actually works and solves the problem. At that point you can start considering whether you might want to give up control and let others join your effort.

The key phrase here is "give up control". Starting an open source project does not mean you suddenly get a staff of programmers you can boss around. In fact, to get it off the ground you will have to be very receptive to suggestions from early adopters and do everything you can to make your tool more useful to a broader audience. And then you give it all away and let people do just about whatever they want with your code. Now you have started an open source project.

SP: What other open source initiatives have you got planned? Given infinite time (and 100 extra pairs of hands) what would you love do?

RL: Well, I didn't plan PHP. I think in terms of solving problems, not in terms of software projects. I actually hate programming, but I love solving problems. So what problems would I like to solve if I had lots of time and resources?

Having recently become a father I had grand plans to build a smart-crib. Live video, audio and various other gadgets that could be viewed and controlled remotely to let remote friends and

family interact with the kid. Of course, once the baby actually arrived finding enough time to just sit and read a book is nearly impossible, never mind building a smart-crib!

Another really cool project would be the ultimate distribution building system. One where I could specify that I had a device with an 80486 processor, a certain type of NIC, and whatever other hardware specifics along with the type of media and ram, and this thing would crank out a small Linux distribution tailored exactly for my device.

Then, to extend that, be able to say that I want it to act as a firewall, mp3 server, browsing station or whatever. Basically get to the point where you can take almost any hardware device and stick Linux on it and have it be useful. I have a lot of devices around the house that I know I can put to better use simply by putting more powerful software on them, and I'd like to find a way to do this without having to spend 6 months on each one.