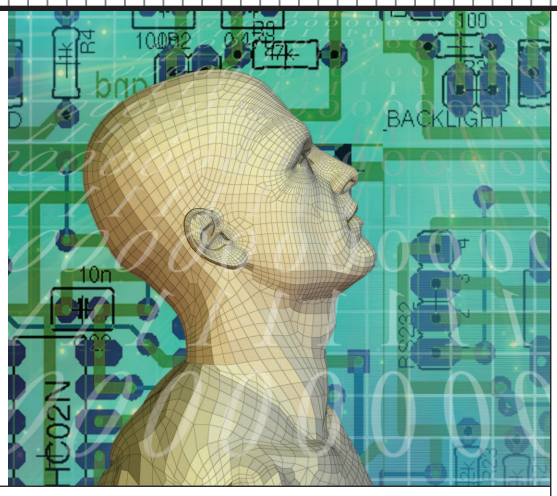
Inventing PHP: Rasmus Lerdorf

Charles Severance
University of Michigan



Unique among most of its peers, PHP wasn't conceived as a pure programming language.

Originally conceived as an HTML templating language, Hypertext Preprocessor didn't start its life as a pure programming language. Instead, Rasmus Lerdorf created PHP in 1994 by collecting the code and utilities written in C that he was using to build websites for various clients:

> I was living in Toronto and doing Web application consulting for a number of companies. I wrote the same code over and over—basically, CGI [Common Gateway Interface] scripts written in C. I wrote code to handle forms, POST data, filtering, and other common Web things that you have to write in C when you're writing CGI programs. It was kind of tedious and boring, so if I could reduce the amount of time I had to spend programming, maximize the output, and get to the solution quicker, then that was my goal with PHP. I put all my common stuff into a C library, hacked it into the NCSA [National Center for Computing Applications] webserver, and then added a templating system on top of it to let me easily call into it.

The first version of PHP was simply a productivity tool that enabled Lerdorf to accelerate his development across his multiple clients who needed Web applications. PHP was quickly embraced by other Web developers, who continue to build on and improve it. To watch the full interview with Lerdorf, visit www.computer.org/computingconversations.

HUMBLE BEGINNINGS

In the Web's early days, the developer community was small, so it didn't take long for Lerdorf's colleagues to find out about his software and start asking for copies for their own clients:

> Other people started asking me how I built these applications, and I said I was using this little tool I built. They asked if they could have it, and I said, "Sure, why not?" My toolkit wasn't what I was selling—I was selling my services of solving problems, and the tool itself is irrelevant, really. It's just my hammer.

After other programmers started using it seriously, they found bugs, fixed them, and sent him patches. Using these patches, he modified his utility library and templating engine and improved the applications he was building for his customers:

> That's when open source really hit me. This was in 1994-1995 before the term "open source" existed. I got together with a group of my peers, other people interested in the Web and solving the Web problem from all around the world. We all faced similar issues and collaboratively we could build a tool that solved our problem. That was really how PHP got off the ground.

Because PHP was initially conceived as a collection of library utilities rather than as a new programming language, Lerdorf never felt the need to shape its future direction. He felt PHP would thrive if he opened the code base to other people and approaches:

> I learned a bit along the way that, for this to grow, I had to give up control of PHP—I had to let other people have some control. I couldn't rewrite patches, both because I'm lazy and it's a lot of work and also to give people some ownership. Once they have full control over their part of it, then they become much more invested in it and passionate. It's not just them contributing to my project—it becomes our project, and that really changed the nature of PHP. This happened around 1997 or so, when I really delegated it out and gave people full access to the source code repository that I was using.

Once Lerdorf allowed other people to become involved in PHP's evolution,

 Published by the IEEE Computer Society

he quickly built a large following around the product:

> The Web grew, and PHP was at the right place at the right time. But also, it was very, very easy to get in and get started using PHP and contributing to it. Even today, it doesn't take much to get a source code repository account in the PHP project. We have close to 1,400 people with accounts, which means those people can all commit to some part of the repository. Slightly more than half the people have committed something in the last year and a half.

The only way to manage all those volunteers is to let them manage themselves. Within the PHP community, many small, dedicated groups work closely together and focus on one aspect of PHP and collectively own it. Lerdorf prefers to let passionate volunteers move forward, even if they make little mistakes that need to be fixed later after their contributions are reviewed by more experienced members of the community.

CROWDSOURCING

Through the PHP Extensions Community Library (PECL), interested groups of volunteers can incubate an idea and then build interest in their feature. Once a feature is in broad use, it can become part of the core distribution, such as the JSON extension in PHP 5.1:

> That's how new features eventually creep in—they live outside of the core tree, get enough penetration and enough people to install them, and then we see Linux distributions pulling them into their core version of PHP. We look at what's happening out there, but there's no real management of that either.

In many open source projects, an individual or small group controls the project's architectural direction to ensure consistency across the product. Lerdorf even leaves architectural decisions about PHP to the community:

> It's a meritocracy. Code speaks. If you write a patch or a piece of code to implement a feature, that says a lot. If someone wants to disagree with that way of doing things, or if they can offer an alternative implementation, that's a really good argument. If all they do is whine about it, that's a really bad argument, and chances are, the implementation will win even though it might not be the best way of doing things. If there's code and it sort of works, that's what we go with, and that has always been the default. It doesn't always lead to consistency, but it does lead to getting new features and actually being able to do something. Being able to connect to this type of database even though it might not be the best way of doing it, at least it gets you there. That's what PHP has always been about—solving a problem. We would rather have an ugly feature than not have a feature at all.

When I asked Lerdorf about PHP's future roadmap, his answer was that it would match the Web's evolution. As the Web moves into new areas and uses new technologies, PHP needs to make those new technologies and approaches available to PHP developers. There's no master plan except to be useful to people developing Web applications.

***Charles Severance**, Computing Conversations column editor and* Computer*'s multimedia editor, is a clinical associate professor and teaches in the School of Information at the University of Michigan. You can follow him on Twitter @drchuck or contact him at csev@umich.edu.*