www.phparch.com

February 2021
Volume 20 - Issue 2

Dealing with Data

Introducing the Mezzio Framework

A Case for Functional Programming in PHP

CQRS And Event Sourcing—Software Architecture at a Higher Level, For Everyone!

Free Sample Article

ALSO INSIDE

PHP Puzzles: Environmental Noise

Education Station: Deeper into the Streams

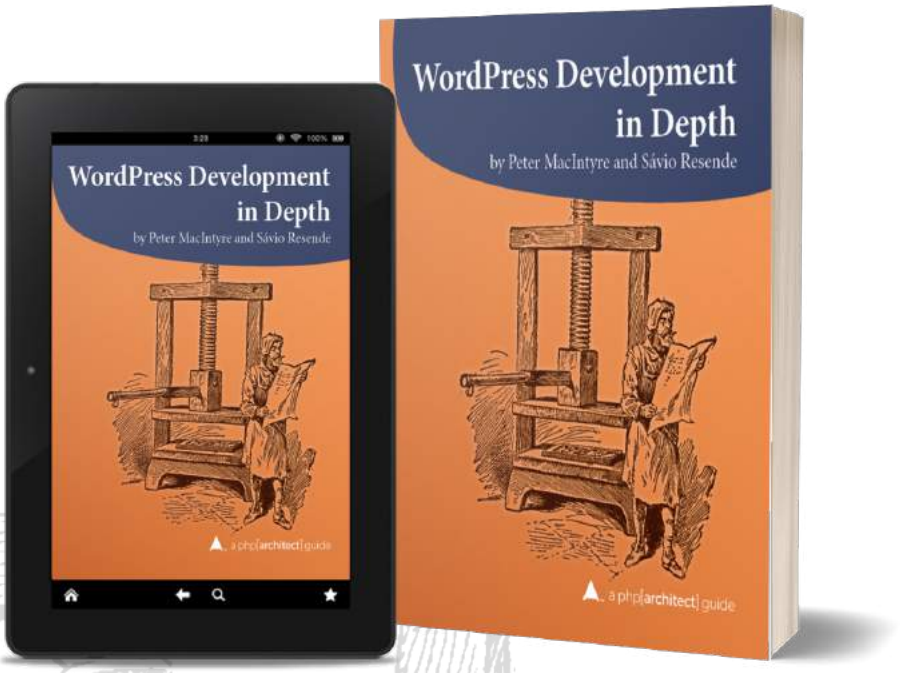
The Workshop: Just Use Docker

Sustainable PHP: Testing Business Rules

Security Corner: Supply Chain Security

Community Corner: Interview with Angie Byron, Part Two

finally{}: Fun with Big Data



Build a custom, secure, multilingual website with WordPress

WordPress is more than a blogging platform—it powers one-fifth of all sites. Since its release, enthusiastic contributors have pushed the envelope to use it as a platform for building specialized social networks, e-commerce storefronts, business sites, and more. This guide helps you sort through the vast number of plugins available to find the ones that work for you. You'll also learn how to use CSS and PHP code to tailor WordPress to your needs further.

Written by PHP professionals Peter MacIntyre and Sávio Resende, this book distills their experience building online solutions for other WordPress site builders, developers, and designers to leverage and get up to speed quickly.

Read a sample and purchase your copy at the link below.

Order Your Copy
http://phpa.me/wpdevdepth-book

Interview with Angie Byron, Part Two

Eric Van Johnson

Now on Drupal 9, the community isn't slowing down. This month, we continue our interview with Angie Byron, a.k.a Webchick, a Drupal Core committer and product manager, Drupal Association Board Member, author, speaker, mentor, and Mom, and so much more. Currently, she works at Aquia for the Drupal acceleration team, where her primary role is to "Make Drupal awesome." We talk about Drupal, coding, family, and her journey throughout the years.

Let's dive into Drupal and, more specifically, Drupal 9. When you run into someone who doesn't know what Drupal is, how do you explain it to them?

I like to say Drupal is three things; it's a content management system doing the things a typical CMS does for you. It's also a content management framework that is built off of a very extensible architecture. So if you are a developer and Drupal can't do something you need it to do, you can extend it very easily through modules and plugins to make it do whatever you want it to do. Finally, it's a community. I've been in the Drupal community since 2005, and we have rewritten it at least three times in that amount of time. The code is completely different from each iteration, but the overall framework is the same, and the people are just amazing. It's a friendly, welcoming, inviting, diverse community.

My elevator pitch for why to use Drupal is: it's good for customers who don't know what they want. You can start it off as just a website, then add a blog. Later, when they need ecommerce, you can add it on, and then when they decide they need a forum with a rating system, there are solutions for that. To build that solution yourself, you would end up with a gargantuan codebase with things stapled together.

How well does Drupal align with the overall PHP Community?

I think, particularly with the advent of Drupal 9, we are much more aligned with the global PHP community. This is where we started adopting some of the PHP-FIG standards for things like coding standards, auto-loading, and HTTP requests. This allowed us to align a lot better with what other PHP developers were doing. We adopted Symfony as our framework of choice underneath the hood so that we could keep Drupal focused on what makes Drupal Drupal, like the Entity system.

We made a deliberate decision back around the 2012-2013 days of Drupal to focus more on embracing the larger PHP ecosystem. We knew it was time to make that change for Drupal to be able to make modifications without breaking everybody's code all of the time, and that was a lot easier with an object-oriented paradigm.

I think prior to Drupal 8, we were pretty much off on our own little island where we would write every single thing ourselves, such as the routing system, menu handling system, filtering system, and other things. With Drupal 8, we really started to look at what was the best of breed for doing things like HTTP requests or access control and using those solutions.

As a developer, what features in Drupal 9 are exciting to you?

I am most looking forward to the automatic updates, which is a major Drupal 9 initiative. This will allow Drupal sites to stay current and patched with security updates. The other thing I am really excited about is the potential of the easy out-of-the-box initiative. Currently, that initiative is taking some features that are not quite part of the standard out of the box experience and polishing off the rough edges, and getting them in. But I think, if we really took that initiative name to its core and we really did make Drupal easy out of the box, I feel that would be transformative and amazing.

One thing I've seen over and over again is that developers seem to love Drupal from a technical perspective. They come in, and they get it. It's sensible, well documented, and has a beautiful architecture. End-users, however, did not have that same experience at all. When you first install Drupal, until recently, you were presented with an empty homepage that just said, "You have no content," and it was very hard for them to grasp the power of Drupal. Now with the out of the box profile, we have a demo set up that makes it so much easier to understand what's going on.

Drupal 9 was the first major release that is backward compatible with its predecessor. Can you tell readers unfamiliar with it more about it and if you think it lives up to the promise of smoother upgrades?

In the past, Drupal had this philosophy of "it's done when it's done" and "we break your code and not your data." What that meant is that we would rewrite all of Drupal between versions five and six and six and seven and seven and eight, and then we do our best to get your data moved over to the new version, but you're on your own for any custom code, your themes, that kind of stuff.

That's a lot of work, and people rightly were very annoyed about this trend, especially when seven to eight move from procedural, functional programming to object-oriented programming and Symfony. This was a big, big change, and people were like, "we're never doing

that again." We were like, "Right, we don't want you to do that again, either." So from Drupal 8 forwards, we've changed things dramatically. We now release every six months a backward-compatible feature version, such as Drupal 8.1, 8.2, 8.3, and now 9.1, 9.2, 9.3. Every six months, you're getting bug fixes. You're also getting new features, but they're backward-compatible so that they won't break any of your existing stuff. Then in a major release, so between Drupal eight and nine, which just happened back in June of this year, the only thing we do is we drop the backward-compatibility layer, and that's it.

We don't make any other changes. Drupal 8.9, which was the last reason for 8 and 9.0, are exactly the same. The only difference is that 9.0 doesn't have the backward-compatibility layers. What it means is that in the olden days, in order to move from one version of Drupal to another, you essentially kind of build out a brand new site and move everything over. Sometimes you would need to figure out what the new modules were for the new version. Now you run a tool that has a dashboard, and it scans through your code, and it lets you know "these three modules need to be updated" or "you need to make a one-line code change here."

So previously, depending on how complicated your site was, a move in major versions of Drupal could be months to upgrade, at very least weeks, but it wasn't exactly something you could just do over the weekend. Now, in Drupal it's much more straightforward, and it should be something that can get accomplished over a weekend.

So the move from Drupal 8 to Drupal 9 is more of an upgrade and not this huge migration, which is really positive and is the plan moving forward.

Does this new upgrade path make it easier to contribute modules?

Yeah, because now modules can work in both versions of Drupal as long as you are using the newer APIs and not the old stuff from earlier versions of Drupal.

How does Drupal attract contributors?

This year has been a little tough because we haven't had traditional conferences. Traditionally we get a lot of mileage out of conferences. We have big events in North America, and Europe called DrupalCon, where developers and users get together in person and talk. They are geared towards the business audience, the contributor audience, and the developer audience. It becomes a weird melting pot of all these different kinds of people.

That is where we get a lot of folks in is through the conference where we educate people who come there to learn something for their job. At the end of the conference, we have what's called a contribution sprint day, which is kind of split into these two areas. In one room, you have a big football-field-sized ballroom with tables, and each table focuses on what they are working on, like media or testing.

In the other room, you have a bunch of people in bright t-shirts running around, and this is the new contributor room. So they set people up with development environments. As soon as you walk into the door, you are handed a USB stick that has everything you need on it, and you are ready to go. They have already prefabricated a bunch of issues that are kind of novice issues or good first-time issues. They pair people up and hand out issues to work on, and then they are around if you have questions.

So we have a track for people who've never contributed before, whereby at the end of the day, they might have their name on a contribution that gets pulled in. The other track is where the developers are more experienced and know exactly what they want to work on; they just need to find the right people to talk to.

It's a lot of fun.

This year we've done virtual sprints, we have issues with novice tags for first-time users. We also have a mentioning channel for people who want to get involved.

Has Drupal benefited from adopting tools from the PHP community like Composer, Symfony, and Twig?

We definitely have benefited. That's the short answer to that question? Twig is amazing in that it allows people to separate cleanly their logic from their presentation and to do it in a way that's interoperable with other PHP projects that use Twig. And that's a skill that you learn once and reuse it a bunch of different times. Adopting a lot of the symphony underlying HTTP framework has been nice because then it's kind of like outsourcing that bit of the code where we trust that stuff's working properly and allows us to stay focused on the things that Drupal can do.

Overall just for the health of the PHP ecosystem, I think it's really good for projects to adopt similar standards and be interoperable with each other because that makes for a bigger community that's able to move around and do more stuff.

We had hoped that move would have triggered a new round of adoption among PHP developers who did not previously use Drupal. If that has happened, it definitely hasn't happened on mass. We had done five years' worth of refactoring to get Drupal onto the new thing. It might be an "it takes time" situation, or maybe people are just happy with whatever technologies they are using. But I do run into those people who say, "I was a symphony developer at my work when they told me now we are using Drupal, and it has been fricking fantastic because now I can combine the two in one," that feels really good.

What does Drupal have to do to compete and thrive in the open-source CMS market?

Like we discussed, I feel making the end-user experience as solid as the developer experience is key. Kathy Sierra wrote about the "I suck threshold", and it's when you use new technology, and at first you're blundering around. You don't know what's, what, and then at some point, you reach the "Oh, I get it" phase, and then you're awesome with it. Drupal's length of time to get past the "I suck" threshold is long, and you really have to commit yourself to it. I feel like that's asking

a lot of people, so I'd like to continue efforts like the starter distro and things like that.

I'd like to continue efforts to reduce that amount of time spent thrashing around in "I suck" and more into the "Oh, I get it" and "this is awesome, and I'm gonna use it" phase. That's kind of going to be my focus for the next while.

What has been your biggest achievement, and what do you feel Drupal's biggest achievement has been as a platform?

For Drupal, that is tough because there are so many different stories. One I am particularly proud of, and unfortunately, it doesn't exist any longer, but during the Obama years of the US government, Drupal was running whitehouse.gov. That was a big deal—not only for us but just for open-source in general. That high profile of a target for nasty people who want to do nasty things was running on open-source was huge.

And that was a huge enabler, not just for Drupal, but I feel like all surrounding open-source technologies. One of the things that the Obama administration built with Drupal during that era was something called "We, the people," and it was a petition website, kind of on the order of change.org. If you got a petition with more than a certain amount of signatures, then that bill would actually go to the floor of the House, and they'd have to discuss it. It was a way of people powering the government to actually care about issues. That made some real actual changes. Making the government more accountable to people and giving people a voice in the government and what types of things that governments should care about. I feel like that was a really neat example of the kinds of things that Drupal can do.

My biggest achievement is being a mom, as cliche as that may sound, but I have a little seven-year-old kid, and she's amazing and awesome and hilarious. She fills my life with joy every day, so that's definitely the best thing I ever did.

It's always such a great reminder that we have these platforms available to us, mostly because of open-source, that are genuinely transformational. Even if we don't program as professionals but code out of passion, the tools, language, and frameworks are used by fortune 500 companies and governments worldwide. We can all add our little pice to make them better, more robust, and more secure.

That wraps up this interview with Angie Byron, a truly impactful and inspirational person to talk to. If you ever have the opportunity to see Angie speak or, better yet, to talk to Angie, don't pass it up! If for no other reason but to tell her thanks for everything she has done for Drupal and OpenSource and everything the Drupal community has brought to PHP.

Eric Van Johnson is the CTO of DiegoDev Group, LLC. A group of passionate and talented frontend, backend, and mobile developers that strive to provide outstanding services. He is also one of the organizers of San Diego PHP (SDPHP), his local user group, and a podcaster. @shocm

Borrowed this magazine?

Get php[**architect**] delivered to your doorstep or digitally every month!

Each issue of php[**architect**] magazine focuses on an important topic that PHP developers face every day.

We cover topics such as frameworks, security, ecommerce, databases, scalability, migration, API integration, devops, cloud services, business development, content management systems, and the PHP community.

Digital and Print+Digital Subscriptions Starting at $49/Year

http://phpa.me/mag_subscribe

php[**architect**]