

Monday, March 31, 2008

### Zend Framework 1.5 And What The Future Holds

I may be tied up (and very close to completing!) a lot of new work that sought me out in late February and has me working long hours without much time to maintain contact with the community, but it's hard to miss the release of the Zend Framework 1.5!

Ever since I started tinkering with the Zend Framework back in 2006 I've pretty much converted to using it in preference to anything else. It's approach in being a PHP5 decoupled component library set it apart so far from everything else in PHP and it was simply easier to work with when you could extend classes to introduce customised behaviour. Of course back then it was still a struggle to keep track of the shifting sands as features evolved, new ones were added, and the integrated whole slowly gained cohesion. More than one custom feature met its death as the features in subsequent versions built up.

With 1.5, everything that makes the framework attractive has quite simply been boosted. We now have a fully comprehensive complex view aggregation/templating system to play with, we have a forms component, translations are integrated as view helpers. Just rattling off several new features is enough to pull together just how comprehensive and cohesive the framework has become over it's 1.0 predecessor. It's that significant. Rather than pondering the shifting sands, I now have a concrete base to build on. Customisations are limited to my own personal branding of how MVC should operate.

It's bloody brilliant!

Obviously I'm overjoyed to finally run out a Core version of Zend\_Layout and Zend\_View Enhanced and cram the custom crap and prior versions into .trash. Zend\_Form which occupied so much of Matthew's time is a marvel in PHP. Without tinkering it's simple to use; with tinkering it can be customised and decorated every which way all in a simple self-contained fashion. Those are my two highlights and it's pretty neat to see the surprise emerging as people finally encounter the Placeholder system for the first time and finally experience everything clicking together. Smarty has never looked so 90s...

For myself, this one is the big release. A lot of developers were surprised and mystified about the original 1.0 release when so many believed it was essentially feature incomplete. Well, no more. The whole package is here and there is very little you can point at as being missing or incomplete anymore.

What's interesting from my own perspective is how Zend and the ZF community at large will build on all this. They have the source code, they have the publicity, they apparently have the resources (they even have Ralph Schindler now ), but can they swarm all over the next few steps to conquer the framework space in PHP?

There are a few ingredients needed to reach the pinnacle of unobtainable perfection and we're no longer speaking strictly of PHP, but also casting a few sidelong glances at the primary competitors for web application development - Ruby On Rails and Django. Personally, I'd worry most about Django since it also has the incentive to pit itself directly against Rails and it has an incoming 1.0 release of its own. Unfortunately, anything connected with PHP faces an uphill battle in a population of Python/Perl/Ruby developers who look down at the most popular web application language because it makes them feel less threatened with extinction.

There are a few strands existing to better the framework judging from the muted whispers infrequently found on the web.

1. Developer Tooling/Support
2. Supporting Emerging Technologies
3. Playing to PHP's Strengths

All three to one degree or another are being worked on.

I could go on forever about the first point, or more specifically tooling, but at the end of the day tooling is all about convenience. Command Line Tools won't make you a better programmer or give you a better application - but they might make life a tiny bit easier. Anyone who've been awed by the Rails tools quickly lose that thrill once they realise the generated files are empty . You still have to type stuff in after. But it's a convenience so many people take for granted,

that when it's missing it makes you sit up and pay attention. A popular missing feature, as they say, is not a good first impression. Importance being totally relative...

Well, try to survive. Wil has proposals and code for CLI tools and generators so it's an ongoing work in progress. Hang in there.

The second part of Tooling/Support is a detour for developers working directly on the Zend Framework itself. I refer of course to my continuing bane - the Proposal Process. I have treated this topic with a lot of attention, and yes, ridicule, more than once and for good reason. It just doesn't work as advertised. I'm been through that thrasher a few times with proposals and it is capable of bringing tears of frustration even to a Guinness-Powered Irishman. It's that bad. The problems have ranged from a lack of resources (nobody to review proposals) to a lack of methodology (how proposals turn into code) to a lack of clear communication. There are other niggling annoyances surrounding control, expertise, and delegation that seem to run aground too - but the first batch are the major ones.

Wil (yes, him again) has undertaken to create a proposal to fix the proposal process after ZF 1.5. I honestly look forward to it. I have a fresh set of proposals which I've deliberately been careful to maintain free of anything remotely resembling PHP code on the basis that if you don't offer code, you won't get nerfed to death by everyone else who assumes you have complete code already written (the anti-Agile/XP pirates).

To point 2, we have emerging technologies. In truth I think the Zend Framework runs the risk of being proclaimed the PEAR replacement for PHP5 which is unfortunate, albeit with a hint of truth. The framework tasks itself with meeting 80% of the common needs of a web application developer - so obviously not a PEAR replacement which has no such restriction. Comparisons aside, the goals of the Zend Framework and PEAR do not agree and hence they are not clear substitutes.

Part of the problem the ZF will continue to face is answering what counts as 80%. Does OpenID count towards an 80% need? I'd wager a firm no, pointing to the accelerating but still scarce on the ground implementations in day to day apps, but then perhaps we should be wagering on OpenID meeting the 80% metre stick in the future. It's a hot topic after all and there's certainly enough mindshare to hedge your bets by including it. So far the Zend Framework appears to accept that risk which is an encouraging sign.

Does that mean OAuth is one of those non-80% emerging technologies the Zend Framework should focus on for future use? I'd wager yes - it's already clear it's being adopted. What about Microformats? Or to mention one I've not personally proposed (for once) how about XMPP (Jabber)? The list of emerging protocols and formats is an unceasing flood that will only accelerate as Web 2.0 oriented applications continue the hustle towards open standards of some description.

This poses another factor of importance - will the Zend Framework be able to attract the expertise needed to contribute implementations? While it's possible to just jump into a standard and implement it, it's probably the more inefficient and riskier option. Someone with both the experience and working knowledge of anything new is more likely to know not just the standard, but that community's interpretation of the standard which is of essential importance since no standard is ever completely fixed in stone. Whoever is most qualified will generally end up producing something that works and is reasonably future-proofed within an acceptable timeframe. I think that's not even half as easy as it sounds - especially in PHP where implementors are scarce on the ground compared to other languages.

On point 3, it's almost inevitable. PHP5.3 may actually end up being a bigger news piece than PHP6 for once. It performs substantially better than 5.2, it has namespaces, there's the funky new MySQL extension and even OpenSSL has been empowered with a swift Diffie-Hellman implementation (if you run OpenID 2.0 using BCMath you probably won't miss those 30-40 second cycle sinks). Tons more I haven't mentioned. While the Zend Framework maintains it's minimum version compatibility eventually that will creep upwards, and hopefully at a pace that doesn't completely return us to BC hell as with PHP4.

In all, the ZF's future is looking pretty good. Even this hardened campaigner for enhanced views, friendly proposal processes, and frequent ranter can't help but feel the future looks rosy for the Zend Framework's future development.

Posted by Pádraic Brady in PHP General, PHP Security, Zend Framework at 16:56

Hi Paddy, I saw this yesterday but wanted to think about a reply for a bit before expressing it in a comment. All the points are well taken, and I'm glad you see a great future for ZF. That's exactly what we're trying to build together! In addition to changes in the codebase for 1.5 that we hope will make the framework that much more useful (I don't necessarily disagree with your comment that 1.0 seems somewhat "feature incomplete"), we're also working on our processes and the ease of use that can be achieved through documentation. You call out the proposal process, and you're right in everything you say. We actually have the new proposal worked out, I'm just trying to find the time to write it up for your and the community's consideration. I

hope it addresses all the shortcomings of the current process and introduces new ways to collaborate with the community from here on out.

I do think we've had something of a change in philosophies since 1.0. Or rather, a change in messaging the philosophies that were already there to get at what ZF is all about more quickly and effectively. The pareto principle (fancy way of saying 80/20 rule) is not a large part of our messaging anymore as we begin to find ways to bring technologies that appeal to smaller factions with the same high quality of other ZF components. We also have de-emphasized ease-of-use as a differentiator for ZF. With the expectations set by RoR and other frameworks, it seems that one would have to implement a framework that reads your mind and write the application for you to offer more value here. We prefer the "no tradeoff" approach: we'll give you the ease-of-use of Rails with no compromise on power or flexibility. In fact, ZF does not- and never has- prioritized what I call the "first hour" of the ZF experience, as we don't feel it is truly representative of the overall experience one is likely to have with a framework in prolonged use. But we're working down our lists of priorities, and we'll get there soon.

I'm really looking forward to moving on your new proposals. They'll be in the first batch we look at now that 1.5 isn't fully consuming our time on the Zend team.

BTW, keep an eye out for a few development process changes as we start to give more developers commit rights while maintaining the high quality we've been striving for since the beginning.

Keep on ZF'in.

,Wil

Anonymous on Mar 28 2008, 16:44

Paddy, have you reviewed the new proposal process yet? You'll find it here:  
<http://framework.zend.com/wiki/display/ZFPROP/New+Proposal+Process+Detail+Draft>  
Please comment on it if there are any improvements you see to be made.

,Wil

Anonymous on Apr 7 2008, 21:22

When everybody tries to implement rails-functionality (generators, active \*, rest, etc.) - why not go with the original instead of waiting decades to make sure it's done (somehow) in PHP?

Anonymous on Apr 10 2008, 22:09

If Rails works well for you, then by all means you should use it. But the fact is that Zend Framework is not trying to keep up with Rails- in fact it has a different set of goals and philosophies. Case in point, we haven't yet implemented generators or active record because a) we don't want to be so convention-based and b) we don't want to reinvent the wheel on complicated components like ORM solutions when there are so many great solutions out there. We'd rather work seamlessly with them. Maybe we'll implement the Active Record pattern at some point, but only when it provides a significant value-add to our users given what's available in the community at large. Why don't you stop by our site and look around so you can see what I mean?

Anonymous on Apr 11 2008, 04:01

Hi Wil,

Will read through it tomorrow - playing catchup on my PHP interests atm

Anonymous on Apr 21 2008, 22:38