

To PEAR or not to PEAR? And how to PEAR anyway?

The title looked better without the verb form



. But really, over the last few months after finally getting over my ignorance of [PEAR](#) beyond it being a hodge podge of packages of dubious quality I've been questioning whether pearifying my future and past code is worthwhile. The answer is a resounding YES.

The problem, if you're not familiar with PEAR you probably understand this, is that PEAR does not have all the coolest libraries. [Swiftmailer](#) is not in PEAR. Neither is Smarty. And look, [HTMLPurifier](#) is not there either. You can throw in others - like the mind numbing lack of a Zend Framework PEAR channel.

Since PEAR only has a few really well publicised packages (like MDB2 for example) it's rarely noted as a first-stop in a search for the ultimate library for your needs. The problem with this lack of attention, after invading the PEAR fortress myself with OpenID For PHP, is that PEAR is actually quite a cool piece of architecture which is about to get even better when PEAR2 goes primetime. Why libraries are not listed there seems to largely be down to perceptions of PEAR having a high barrier to entry. I don't use the Economics term lightly



The perceived barriers to entry include:

1. PEAR now only allows PHP5 in new packages

While this might look bad to some, let's face it. PHP4 is ancient news. If you are a developer, and still using PHP4 only for new code then you have lost the plot. I'd only forgive you if you used PHP4 expressly for the purposes of supporting down on their luck users on shared hosting not offering PHP5. But that excuse is getting thinner every year at this stage.

2. PEAR will require large scale changes to my shiny new cool code

PEAR encourages two development goals. Adherence to a popular recognised coding standard - predictably called The "PEAR Coding Standard", and adherence to sound design principals. Members will quickly find and expose design flaws, criticise your cumbersome API, and flambast your lack of a maintainable design. On the otherhand they will readily praise the presence of tests and docs, and the sight of a well designed library. PEAR is like the ultimate QA Process - and nothing to be afraid of if you are already producing good source code. Besides, if you can't accept simple criticism you should stop developing because it's unavoidable in open source. Criticism in PEAR is a healthy constructive slice of feedback which should be welcomed with open arms. It only requires huge time consuming changes if your code, is arguably, a pile of spaghetti to start with.

3. PEAR only allows proposals for complete functional code

PEAR2 will soon be here which means all you happy XP/Agile folk can eat all the cake you wish. I'm ambiguous here myself as regards to the old steadfast PEAR. Maybe it's the legacy of an existence before Agile practices became popular, and the code was all important, but I believe most PEAR members are not living in the dark ages



. The Proposal system is not Agile oriented at present, so if you have a new idea visit the mailing list and ask about proposing before code exists for better advice than you'll get from the official

manual. PEAR2 sounds like it will formalise more Agile friendly means of gaining proposal acceptance and support for developing from scratch.

Of course "complete" is an exaggeration. Proposing an alpha or devel package to PEAR as it stands seems pretty straight forward. Just do the startup TDD and coding to get to some level of functionality before proposing. Then improve over subsequent iterations as usual to reach a beta.

4. PEAR is elitist

PEAR's social norms are often criticised. However few people seem aware that PEAR has suffered a recent revolution and is now governed by a democratically elected PEAR Group and President. Vive la Résistance, mon amis! As for the elitist front, I've never seen any sign of it. The only thing I've experienced from the mailing list and private emails/IMs is a sense of a helpful friendly community of developers who are more than aware of the past problems of PEAR and who are motivated to fix them. If anything folk seem to be curious about my experience in PEAR and are readily open to feedback on how PEAR works. Seeing as how I'm a total n00b to PEAR that speaks for itself about how non-elitist it really is.

5. PEAR code sucks!


Indeed. It's hard to argue that some proportion of PEAR packages are plain awful. But given how many packages PEAR has (my first CVS checkout was an interesting experience for a novice), and their age, that's inevitable. I like to think of PEAR as a function of software development progress in general. As development practices, PHP tooling, and PHP features progressed, so has the code on PEAR. With PEAR2 around the corner (even more progression!), the opportunity for renewal is literally almost upon us.

6. PEAR is fossilised

See 5. I haven't been in PEAR long enough to know how fossilisation is viewed by the community. But I think it's obvious you can't pin this completely on PEAR. Any library of sufficient age and penetration is going to fossilise. I think what's important is that a path for proposing alternatives with duplicate goals is allowed by some set of criteria. Over time any fantastic library which doesn't innovate will die and shrivel as a younger innovative library stamps all over it gleefully with a fresh approach and superior mindshare. I look at PHP Mailer for example, and then Swiftmailer, and wonder when PHP Mailer lost its will to survive.

7. I'm not going to contribute to PEAR, ergo it's useless

So wrong. I too was so ignorant. Then I met this weird thing called "pear package". Creating public or private PEAR packages of your own libraries is such a gift to PHP. No more complex copy and text replace tasks, no more fiddling with include_paths (whose every extension slows down your application), no more fizzy installation instructions. If you have never created a PEAR package, or don't know how to install a PEAR package you (or PEAR) can offer, then you are missing a treat.

For those with an affection for build tools, like Java ant or PHP's own Phing, you can easily script a PEAR package task. I've started using the d51PearPkg2Task for Phing from Travis Swicegood and it's such a simple piece of automation to have on hand. PHPSpec will now be distributed as a normal tarball download and as a PEAR package from some channel when released. I just run "phing dist" from a command line and it magically appears 

Okay, so maybe you won't contribute to PEAR. But PEAR's tooling is hard to ignore. They have the only decent distribution method supporting automated installation in PHP. If you don't take advantage of that then it's your loss. And your users' loss.

8. I can't use PEAR on a shared host!!!

Yes, you can. The secret is in setting up your `include_path`, or using `__autoload`, or finding one of the tutorials for doing it. Any shared host can support a custom PEAR collection with a small amount of effort which will override the default (out of date) version your shared host currently provides.

I'll answer the "How to PEAR?" question in another entry soon. If you feel one of the 8 (undoubtedly there are others) barriers above personally, I suggest you look a bit more closely at PEAR. Evolution is in play.

Posted by Pádraic Brady in PHP General, PHP Security at 05:43

I've got to say I've been guilty of believing some of the PEAR myths . Next time I need some code I'll give it another look.

Thanks for your time writing this post. Anonymous on Oct 20 2007, 06:31

I came to PEAR in 2006 for a Google Summer of Code project. Since then the community gave me all the support I needed. PEAR introduced me to the Open Source world and teach me how powerful the development with a friendly group can be. Anonymous on Oct 20 2007, 11:57

Well, I did think the same, until I found PEAR Calendar. It is too bad it isn't maintained anymore (or at least it hasn't been for a while). It is the best calendar class library I've been able to find. There are some things I would do differently, but it just saved so much time.

From the perspective of PEAR I would like to know how it could be used to install full applications. Without the `include_path` option, I wouldn't have thought it would have been possible. I'll need to look further into that. If it will be possible, I think it would be the best thing ever to just say, pear install WordPress/WordPress and let it go.

I've been doing some basic research and development on that basis. PEAR2 looks damn awesome in my opinion. Anonymous on Oct 21 2007, 03:14

You know, the worst thing about PEAR is that its own core code sucks. We had been integrating pear package management into our product this summer - it has taken more than 180 hours to do so.


Who in their right mind would extend PEAR_Installer from PEAR_Downloader?

Looking forward for PEAR2. Hoping it would not suck that much. Anonymous on Oct 21 2007, 04:13

The worst thing I had experienced with while using PEAR is tons of warnings with `E_STRICT` mode.

Thank you for article, I'm about to check whether things are changed. So, maybe this time... Anonymous on Oct 21 2007, 18:35

@Jacob: It depends on the application, but if you take a Zend Framework example where the application can be stored anywhere with just an `index.php` pointing to it, then it's easy enough to distribute via PEAR. The only additional step would be a post-installation script to write the relevant `index.php` file to the web root.

@Weirdan: Hey man 

. The one danger is that PEAR2 becomes a lazy fest, with PHP4 packages moved to PHP5 without any additional design updates, refactoring or such. PEAR2 has a new core system called Pyrus which is in subversion on the PEAR2 repository - I'd suggest giving it a quick review for your sore points with the current PEAR and get your feedback to the PEAR mailing lists. The early bird...

@Alex: PEAR has tons of PHP4 code so I really doubt we'll ever escape `E_STRICT` warnings. PEAR2 which is PHP5 only will of course be `E_STRICT` clean. PEAR2 will not allow any PHP4 code. Anonymous on Oct 21 2007, 18:53

Pear SUCKS!

Proof? Show me 1 website that shows Pear components. Delphi or Microsoft GUI components are all simple drag drop. Pear does not even show what the components look like.

Pear is command line. Command line is inefficient 1 dimensional. Graphical is 2 demensional efficient. After months of archaic command line monkey, twisted research, the competition has simple intuitive GUI component selection. UGH! Anonymous on Nov 13 2007, 02:42

@Bill Johnson: you simply don't understand what you're talking about. Packages in pear do not 'look like' something. They do work. They are mostly libraries, providing some functionality. And no, DB abstraction layer does not need any icons, graphics and such. Nor does cryptographic library. Anonymous on Nov 13 2007, 03:30

Learn the difference between a library and a GUI component, or maybe actually use PEAR before commenting such nonsense. Anonymous on Nov 13 2007, 04:26