What's Wrong With ColdFusion? Round 1 Response

Posted At : September 30, 2013 10:06 AM | Posted By : Cutter
Related Categories: Development, ColdFusion, Adobe

Last week I **raised the question**, and Wow! Have people responded? Yes. There's some fire in the belly, and a lot of people have a lot of opinions. More importantly, they have a lot of great ideas as well. So, what I'd like to do here for a minute is break down all of that feedback, to get down to the meat. We've attracted some attention, with our little discussion, so I think it's important to focus on all of the core points. From there? Good question. People are listening. Maybe we help to blaze a new trail in the wilderness.

Make It Better

First and foremost, there was an overwhelming call to "get with the times". Among complaints about stability, setup, and pricing, the key take away was that the server needs some retooling. Smarter people than me have talked about the pitfalls of CF running as a servlet, issues with context roots, and (continuing) issues with object instantiation, and all of these things should be addressed. Take that feedback and couple it with the two largest suggestions in the bunch: Modular feature deployment/package bundling and management, and a Command Line Interface (for multiple reasons).

A modular approach can absolutely open a lot of doors, both for developers and Adobe. It can allow someone to deploy a server with only what they require for their application, or upgrade that server's capabilities as required as well. These cut out the cruft, allow the system to be smaller/faster/better, and possibly provides a new (and ultimately more profitable) way for Adobe to handle the licensing of their product. Strip 'er down to her core, then provide deployment packages for things like cfform, orm, office and exchange integration, etc. This modularity also would allow Adobe to laser focus on updates to key areas in a more timely fashion as well (like bringing SharePoint or Exchange support up to the current version in a productive way, etc).

As a broader part of that picture, many asked for the removal of features (UI cruft, cfform, all tags, etc). This isn't really practical, considering the thousands of applications running and supported on ColdFusion today. Adobe needs that licensing revenue. That said, the revenue goes away if they (insert business names here) start switching languages. This modular approach opens new licensing models, better tailored to larger business (and developer) needs, and can address key concerns on the cost of scale. The comparison to the current cloud offerings are a prime example of how this model could truly be a more effective tool for everyone.

The next major suggestions all revolved around core language improvements. The

tag model does have it's place, especially with the templating aspects of ColdFusion (generating output), but CFScript, and associated core "functions", require some overhaul. There's a lot of inconsistency in naming conventions, argument order and treatment, lack of member functions, core output, etc., and there's some serious call to correct that. CFScript should be more ECMAScript compliant, and Adobe already has serious experience with this with their development of ActionScript. How they do all of that, and maintain backwards compatibility at the same time, is a question for folks smarter than me, but it absolutely makes sense. Working in ColdFusion should feel native to anyone who writes HTML, CSS, and JavaScript for a living, and right now it's a little off the rails there, especially when it comes to script.

True Issues To Address

Aside from the suggested improvements, there was a lot of conversation about things that truly are broken, and require attention. I'm not just talking about the list of bugs in the bugbase, but some core process issues. Adobe ColdFusion is a paid product, and that licensing includes "enterprise" support. As such, a common complaint is that the support is unknowing, uncaring, or unresponsive. When I call in for support on install issues with Creative Cloud, I was able to get quick and solid resolution, and follow-up. Why should ColdFusion be any different? And why, when a true issue is brought up (like some of the recent security issues), should it take weeks and months to admit the problem, allocate resources, and write/test/deploy fixes?

I, and everyone else, understand that ColdFusion server development is a closed ecosystem, being that it is a corporate product. It does not have the benefit of rapid development that an Open Source project might have, with a large community behind it. That said, considering the costs involved in licensing the product, the turnaround model on identified issues actually ranks a higher priority. How you interact and respond to your customers is critical. This is just business sense, as the customer you keep happy is the one that you retain and keeps buying your product (and tells others how awesome it is).

The other addressable issue, brought up in the conversation, is marketing and education. ColdFusion isn't being taught (or it is, but very little), and new developers aren't coming in to the fold. Some of this is marketing, some of this is perception, and some of this is ... Well, what is it? Why isn't every high school and college in America (or the world) given licensing for ColdFusion? Not ed licensing to dev on, but actually running their university sites, and student unions and such? Why aren't there coding challenges being sponsored in classrooms, to help create the next round of rockstar open source applications? And why aren't there timely, up-to-date, curriculums being developed for these markets? Materials that help teach the language, with focus not only on the language and the server but also development as a whole and best

practice?

OK, so all of these "issues" require resources, in an age when things have been scaling down. We've identified them as issues. We've identified their importance. Now it's up to someone to decide the value of addressing those issues.

Where's Our Community?

ColdFusion has, for me, always been a great community. There are hundreds of developers out there who have worked at this stuff for a long time, tripped over good code and bad, and attempted to provide their own brand of support through blogging and answering StackOverflow questions, and just sharing in general. They've done these things, usually, on their own time, with their own money, and at the sacrifice of other things.

But, it's easy to fall off that wagon when you're trying to help others and someone else just tells you "you're doing it wrong". A great case in point here is the OO debate: should you or shouldn't you. First, I'll say "yes, you should", for a thousand little reasons, but then I'll also say "you don't *have* to, but you *should*." Second, I'll say that there's no one way to approach development, and that each approach provides it's own pluses and minuses. The single greatest thing that you can do, as a developer, is to read other peoples' code, and realize that it might not be the best approach, or that it might be a much better approach. Each of us has something to learn from someone else, as each of us has a different perspective. Yes, sometimes that perspective may be wrong, and it's good for us as a community to police one another, in our examples, to help promote better code. It's **how** we have those conversations that will change us. "This Sucks," is not a productive response. Sure, I hate getting called out on my crap code, and then I learn from it (thanks Ray ;)) [For the record, go read the code for FW/1. Outstanding! The things I learn from Sean...]

The other side to that coin is the framework debate. We have a lot of "mine's bigger than yours" going on, still. The thing is, these are tools, each of them written differently and from different points of view, and if all of these framework developers were to collaborate, instead of pulling out the yard stick, we could really move mountains. You people are smart, and you're dedicated, and you have great ideas. Please work with each other instead of against one another.

And this comes down to the final area of discussion, in round 1, and that's our community's willingness to share their work. There are many people out there that write thousands upon thousands of lines of fantastic code that never sees the light of day outside of a firewall. One great comment I saw said something like "What if every CF project were out there and you could just drop it in and it almost worked?" How

many user management modules have you written? Permissions systems? Rudimentary document management? Can't you put anything on GitHub? RIAForge? Every small bit of code we contribute, as a community, adds to the overall ecosystem. We have to start helping each other solve common problems. These other communities, that so many developers are turning to? That's what they've been doing right. And that's where we, as a community, have been failing. There were a lot of fingers pointed toward Adobe, in the comments to my post, but I'll remind you that every pointing hand has three fingers pointed back. We have brought a good bit of this upon ourselves as well, and only we can change that.

I, for one, do not think the battle is over. I, for one, still think there's a life for ColdFusion and ColdFusion development. Yesterday Matt Gifford and Scott Stroz and I sat down for the CFHour podcast, and Scott said "If JavaScript can have such a resurgence, then why can't ColdFusion?" And he's right.

So, this recaps (very high level) all of the comments on my last post. The discussion has begun, now where do we take it? Can we define a path for ColdFusion's future? Is Adobe part of that discussion? (I hope that they are) I would like to hear from you again, dear reader. What are your thoughts on what's being said today? Throw out the negative, focus on solutions, and tell me what you believe can/could/should be done for us, as a community, to move forward and thrive.