What's Wrong With ColdFusion - 2014 Edition

Posted At : October 13, 2014 6:15 PM | Posted By : Cutter
Related Categories: Development, ColdFusion, Adobe

Well, it's that time again. Later this week is the second annual ColdFusion Summit, put on by Adobe. Last year, just prior to the summit, I posed the question **What's Wrong With ColdFusion**. The feedback was **immediate, and intense**, with many people weighing in with their thoughts on what could be done to progress ColdFusion, both as a platform and a language.

Unfortunately, last year I wasn't able to attend the summit, and development of the ColdFusion 11 server was nearly complete, so most of that feedback had zero impact on the new release. While their were many advancements in security and JSON processing, some of the other introductions were seemingly underwhelming, or even unwanted, to many long time CF developers.

Now, to be fair to Adobe, and the ColdFusion product development team, they do have a responsibility to cater to their paying customers. Developers are rarely the ones buying the server itself, and the money handlers buying the server licenses rarely have enough real understanding of the development process to truly provide reasonable feedback when asked "What do we do next?"

Past issues aside, the ColdFusion product team is beginning to explore what they will do with the next version of the ColdFusion server, so now is a good time to begin the discussion again as to what Adobe might do the next time around. It is a delicate balance, so our feedback to the team needs to be positive, constructive, and backed by hard arguments to support our requests.

ColdFusion has been known, since the very beginning, for making the hard things easy, with it's first inroads being in the are of data connectivity. ColdFusion makes it extremely easy to connect with a wide range of datasources. This was probably the first true task of CFML (originally called DBML). It's meeting this type of root level need that ColdFusion might need to focus on going forward.

There were a number of core concepts that came out of last year's discussion:

- Script Language Overhaul
- Command Line Interface
- Package Manager
- Modular Architecture
- Support for NoSQL Databases
- Application Profiling

There are some great suggestions here, and Rupesh actually had included these in his final slide of the cf.Objective keynote presentations as possible items of inclusion in the next version of the ColdFusion server. I, personally, say to Adobe that it is important to review the past year to help define the focus for the future. It is important for the product team to not waste time on solutions that are already available, or that developers can already do themselves with a little study. In that vein, the CLI and Package Manager may not be areas for Adobe to spend time on, since we now have **CommandBox** available to us. Focusing on solutions that are already freely available to us seems to be a waste of time and resources, unless you intend to go above and beyond any available solution. A key question to ask, in this type of evaluation, is not only "is this already available?" but also "does this truly fit the purpose, enhancement, and growth of our product?"

Modular Architecture

So, what sort of things can Adobe do? Well, the Modular Architecture concept has multiple levels of benefit, not only for developers but for Adobe as well. By creating a "core" build of ColdFusion (db interactivity, includes, custom tags and the like), with "packages" of additional functionality (ORM, CFFORM, CFCLIENT, etc) that could be installed only when required, Abode creates an architecture where developers can customize their environment to the needs of their application. This decreases the overall footprint of the server, removes the "cruft" that might be unnecessary, and expands the licensing options for Adobe as well. I also believe this could significantly improve the cloud options that could be offered for the server as well, which could greatly improve adoption of ColdFusion as a platform.

Image Processing

Image manipulation was a long time coming to ColdFusion, and was a great addition once it finally arrived natively. That said, the current implementation is very process intensive and slow, and support isn't quite what you would expect from a brand like Adobe. It may be time to revisit image processing in ColdFusion, possibly even exploring avenues other than Java for handling these processes, to improve quality and performance.

Server Performance

There is strong evidence pointing to other CFML engines (Railo) as having much better overall server performance. It is difficult to allocate time and resources to something that currently works. But, just because it functions doesn't mean it shouldn't be improved upon. Response time is a critical piece to any application, and Adobe ColdFusion (as a paid and licensed product) should be the best of breed in this category.

Language Overhaul

Again, it's difficult to put time and effort into something that already works. But, the current syntax of script and functions within ColdFusion **is** a barrier to adoption of the server. This may be one of the biggest, and most asked for, enhancements that Adobe could provide to ColdFusion. Developers, who have traditionally worked in other languages, spend little time evaluating ColdFusion as an option because of the belief that ColdFusion is a second class language. Why would you use ArrayAppend() when every other language uses array.push()? Many objects do provide Java level object access, but many of these are undocumented, or aren't displayed in the documentation as the first line of how something should be done. While ColdFusion must support the current language constructs for some time to come, it really is time to make a more ECMAScript compliant form of script for ColdFusion. Or, at minimum, something much more uniform and familiar.

What Next?

So, with the summit starting later this week, I pose the question again "How can ColdFusion be better?" What would you like to see Adobe focus on for the next version of the ColdFusion server? And why? How? To what purpose? What do you think are areas where Adobe should absolutely steer clear of? It is important that we provide our feedback now, early in this process, and that we do so in a positive and constructive manor.

I will bring any comments given here back to the ColdFusion product team at the summit later this week (yes, I'm presenting this year). I encourage everyone who is going to the summit to voice your opinions directly to the product team as well. I look forward to hearing everyone else's input on this.