

RE: Java haters, gtfo [radio edit]

Well everyone's [favorite potty mouthed blogger is back](#), slinging poo and doing nothing much to help anything. That said, I've met Hani, and he's actually a pretty cool and down to earth guy. So with all due respect, I'll respond in kind with language Hani can understand. I fully appreciate the satire on his blog, but this is the first time I actually feel compelled to respond to it.

Here's why:

I fear that anyone who agrees with him is at risk of becoming a mindless 9-5 Java drone, who's willing to follow Sun off of a cliff.

Sound familiar? It should. Because it's what so many Java developers have criticized Microsoft for in the past.

So on to Hani's first question:

"Why are so many people angry at Java?" ~Hani

We're not mad at Java, that would be silly. We're mad at the steward of the Java platform, the leadership. Here's why:

Sun has been trailing behind "current" and failing to listen to customers for the better part of 8 years now. By trailing behind, I'm not talking about being "bleeding edge". I mean reasonably current. It's not just the language, but we'll get to that in a minute.

First, let's look at Sun's track record for evolving software in general:

- EJB had us generating code, stubs and writing vendor specific deployment descriptors to achieve what? Meanwhile Spring trounced them became an alternative way of leveraging the useful bits of J2EE/JEE. It did such a good job that it actually threatened the existence and need for application servers (BEA even released WebLogic Express sans EJB support). YEARS later, Sun responds to Spring with EJB3, by which time Spring is so popular that it is quite literally "the new app server". Good on Rod for standing his ground.
- CMP and JDO were DOA; completely failed out of the gate. Meanwhile Hibernate took the world by storm, laying waste to open source and commercial frameworks alike. YEARS later, Sun responds with JPA (a spec), which is basically a carbon copy of the intersecting functionality of TopLink and Hibernate. Shame on the Hibernate team for taking part.
- Servlets and JSP were severely lacking in the framework department, the gap which Struts filled so many years ago. At one point I heard a stat that claimed that 70% of Java web applications used Struts. So Sun hires Craig and has him build a spec that is released YEARS later: JSF. And what a dog that was. They couldn't even succeed when they hired the same guy to do it over again. Shame on Craig for taking part.

[Edit: I am not implying here that Sun SHOULD Be building these frameworks. In fact I'd prefer they stay out of the framework and 3rd party specification business. Eg. Hibernate was better before JPA came along.]

Does this pattern sound familiar? Again, it should. It's called the "embrace and extend" model, made famous by Microsoft.

The only difference is that Microsoft is good at it, and Sun is not. The "extend" part is usually where people criticize Microsoft for their incompatibilities. In Sun's case the "extended" part is usually in the form of something LESS functional, but that they'll ultimately call a "standard". It doesn't do anything different, it's just that Sun's way is "standard" and the other is not.

Here is Sun's greatest failure:

What have these standards and specs bought us in the past? Ah yes, a tie to an app server vendor, inflexible code and stuff that breaks if you try to port it to a different app server. **Exactly the problems standards should solve.** Meanwhile, the Struts/Spring/Hibernate apps all ported without issue. What?! The nonstandard app is MORE portable than the standard app? If that isn't Sun's most classic failure in history, I don't know what is. The thing is, they've repeated it, and they continue to do so as long as they write specs for app servers.

By the way, where are those standards now? Did they save you any time, money or were you able to just "plug-in" to upgrade? The number of people still running JDK 1.4.2 on WebSphere tells me that the standards have failed them.

Anyone else tired of chasing Sun's tail?

Time after time, Sun continues to respond with far too little, far too late and with lackluster results. By the time they've caught up, nobody cares and the world has evolved around them. Perhaps a flicker of life left in Sun is that they've chosen to invest in JRuby and (to a sadly lesser extent) embrace Groovy.

This is a rare turn of events, because Sun rarely responds with actual software. Instead they respond with artificial specifications that end up costing everyone from software vendors to customers MILLIONS of dollars, because they sling the word "standard" around and whip everyone into submission with their licensing agreements.

Anyone else wonder why there have been 10 major versions of most app servers in the last 10 years?

That's "major" as in "hard and expensive to upgrade, even to a X.1 release". How many have you been able to seamlessly upgrade to? How many have been free to you? How many have you avoided upgrading to, even if it meant you were stuck with *JDK Flinstone Edition*?

JDK 5 aside, any Struts/Hibernate/Spring software will still run on any version of any app server. For how many J(2)EE standards-based apps can that be said? I suppose with a vendor specific deployment descriptor...

And NOW this is same kind of unhelpful evolution is happening to the Java language itself, which I'll call:

The Java 5 Bolt-On

We used to be able to ignore all of the above specs and just build better frameworks using a fairly consistent, simple and predictable language. Sure, it wasn't feature rich, but we've never complained about that (or at least I haven't). What I care more about is the evolution of the language, and I feel that Java 5 set a dangerous precedent.

It started the trend that bolt-on evolution is okay. Rather than a good, hard, major upgrade to the language, they added on a bunch of half-baked features and tried to make the current Java something it's not. They were in a PR war with Microsoft and thus were trying to copy what C# had already done, but with an artificial schedule and no money.

They failed dramatically. All this while waiving the flag of "[backward compatibility](#)" that they also ended up failing at anyway. Think about this! What would be the cost of implementing generics and annotations correctly (see C#), compared to upgrading from EJB 2.1 to 3?

Java 7 will repeat these mistakes, but this time they're copying Ruby.

Sun does not understand short-term sacrifice for long term gain. Instead they choose short-term PR and marketing over any gain whatsoever. Sun chose poorly. I won't rehash all of my Java 5 thoughts here, but you can read them [in my other posts](#). Or, I'd be MORE than happy to have a live debate with anyone on this subject.

This is a far cry from [2002 when I publicly advocated Java](#) as far superior to .NET (with code) and 2006 when I debated on a stage against Dave Thomas and various ThoughtWorkers on the subject of outrageous Ruby claims.

So in all fairness, of course I'll use other languages as a point of reference when discussing the evolution and future of Java. This brings us to Hani's next statement:

"The first glaringly obvious point of commonality is that their new language is something with low adoption (compared to Java), is fashionable (this month/year), and is filled with ex-Java people." ~Hani

Hani: I don't need to talk about Ruby or Haskell or Scala or Groovy to be mad at Java.

I can compare Java 1.4 to Java 5 and be mad. Or I can look at what's forthcoming in Java 7 and be mad. In this regard, YOU of all people with your "get off my cloud" attitude should be mad. Why? Because it's not ME changing your language. It's Sun and the JCP. And they will turn it into something you hate, mark my words. Whether dedicated Java Developers will be able to admit it to themselves is another story.

My personal disappointment isn't the idea of change. It's the quality of the changes. And for comparison here, I look to Microsoft C# (not Ruby or Groovy).

The Microsoft team is doing a bloody awesome job of evolving C# as a language and making changes that developers actually need and want. Even LINQ (which Java developers still think is embedded SQL) is a brilliant addition to their language, and the way that it's implemented is elegant and consistent. I personally am more interested in the new individual features they added to support LINQ.

I don't envy C# itself, but the C# team and their leadership.

They used neat concepts like pattern recognition to add new keywords without breaking backward compatibility. That way, they didn't end up with an "@interface" instead of "annotation" because their compiler could actually tell the difference between the contexts in which those words are used.

There's at least 10 (similar) significant improvements that could be made to the Java compiler that would clean up the language with no backward compatibility issues or JVM changes. **But I fear Java 5 has dug a hole that's nearly impossible to get us out of now.**

Don't get me wrong, there are features of C# I don't like. But they're more easily ignored and implemented consistently enough that if I do ignore them it doesn't cause chaos within the source code. For that matter, Visual Basic .NET is looking fantastic and a lot of very smart people (who despised VB6) are looking more favorably at VB9 than even C# 3.0.

Microsoft is generally doing a better job of managing their platform than Sun is. In the case of VB, **their most popular language**, they had the courage to cut the cord and make something better!

"The second point is that there has to be some barrier to entry. It can't be something that's generally useful for which you can hire easily," ~Hani

C#. Within 5 years, you will have to cross an ocean to get a job doing anything other than maintenance with Java.

*"Here's a novel idea, how about getting a job and shutting the **** up about it?" ~Hani*

Got one.

Here's the difference between my job and yours. I'm a Developer, not a Java Developer. I've had actual real-work experience with Java, C# and Ruby (and others before Java). I can give you a fairly detailed breakdown of why they're all great, or why they all suck. Indeed, I'll give you my reasons why I don't think Ruby would survive another 5 years if Ruby developers were honest with themselves (and why they'll pay for it later). Or, I can give you 25 things I'd change about C#.

But the one that I'm the most disappointed about is Java. Here's why:

Java is an unnecessary failure. It doesn't need to suck. It is only for a lack of courage, bad leadership and bad decisions (and perhaps no money) that it sucks.

Could someone else have done a better job? Perhaps only for one reason: Anyone could have done better by doing NOTHING to the language and staying out of writing specifications for app servers and then dare to call them "standard". Less would have been more.

*"If your life is so great, why the **** must you CONSTANTLY hassle us and **** in our coffee?"~Hani*

Because I'm Sun's customer too, and I've got a 10 year career investment in their bitter coffee. So my dissent is warranted.

The difference between you and me is that I'm not interested in spending the next 10 years following Sun's misguidance.

Cheers,
Clinton

DISCUSS: At DZone...